UNIVERSITÉ DE MONTRÉAL

AN IMAGE PROCESSING APPROACH TOWARD A VISUAL INTRA-CORTICAL STIMULATOR

ANTHONY GHANNOUM
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
DÉCEMBRE 2012

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

AN IMAGE PROCESSING APPROACH TOWARD A VISUAL INTRA-CORTICAL STIMULATOR

présenté par : GHANNOUM Anthony
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. BRAULT Jean-Jules, Ph.D., président
M. SAWAN Mohamad, Ph.D., membre et directeur de recherche
Mme CHÉRIET Farida, Ph.D., membre

ACKNOWLEDGEMENTS

I would like to start by thanking my Master thesis supervisor professor Mohamad Sawan for his support and patience as well as the Polystim Neurotechnologies Lab members who always provided a lighthearted mood and critical thinking environment.

I would also like to thank my sister Roula Ghannoum for her outstanding help throughout this period.

Moreover, I would like to acknowledge support from NSERC and the Canada Research Chair on Smart Medical Devices.

RÉSUMÉ

La déficience visuelle n'est actuellement pas médicalement traitable. Par contre, des approches biomédicales modernes, tel que le projet Cortivision, emploient la micro stimulation intra-corticale pour stimuler la vision électriquement. Cela fait apparaitre des «phosphènes», des points lumineux, dans le champ visuel du patient. Des constellations de ces phosphènes peuvent être utilisées pour reproduire une certaine vision de base.

Le système comprend une caméra, un module de traitement d'images, un lien sans fil, et des matrices de micro-électrodes implantées directement dans le cortex visuel du patient.

La carte phosphène (l'arrangement de point lumineux dans le champ de vision) est directement liée au placement physique des électrodes ; elle est généralement plus dense vers le centre (plutôt que d'être espacée uniformément comme dans une image numérique). Cela veut dire qu'il n'existe pas une fonction simple pour traduire les images acquises par la caméra sur la carte phosphène, d'où la nécessité d'avoir un module de traitement d'image.

L'aide du patient est nécessaire pour déterminer sa carte phosphène particulière et réelle. Une méthode de calibration qui expose le patient à des combinaisons de paires de phosphènes différentes est utilisée avec des algorithmes de triangulation et minimisation d'erreur. Toute erreur dans la phase d'estimation induirait des distorsions dans l'image finale (dans les stimuli envoyés au cortex du patient).

Néanmoins, la plupart des systèmes utilisent une carte phosphène simplifiée uniformément répartie ainsi que des techniques simples de traitement d'image comme un seuillage simple ou la segmentation. Nous proposons d'utiliser un traitement d'image de plus haut niveau, comme la reconnaissance d'objets, afin d'être en mesure d'identifier et de «simplifier» les données avant de les envoyer au patient. Dans ce cas, nous envoyons une représentation du contenu de la scène au cortex du patient, et non pas une version abrégée de la scène – l'algorithme utilise les vraies images pour identifier les objets, tandis que dans le cas des autres systèmes, le patient doit identifier les objets lui-même à partir d'une image très simplifiée ; une tâche non-triviale. Une telle approche est coûteuse en terme de calcul, et nécessiteraient également de rouler en temps réel pour notre application. Pour subvenir à ces besoins, nous proposons d'accélérer les goulots d'étranglement de l'algorithme en utilisant une approche matérielle parallélisable (FPGA).

L'objectif est de développer un système temps-réel, paramétrable, et efficace en termes de ressources matérielles FPGA et de bande passante. Ceci nous permettra d'effectuer des tâches de reconnaissance d'objet robustes rapidement qui peuvent ensuite être utilisés pour stimuler la carte phosphène d'une manière plus adéquate, plus utile, et moins exigeante du

patient.

Après avoir examiné les algorithmes possibles et les approches qui peuvent être utilisées, nous avons identifié le Scale Invariant Feature Transform (SIFT) comme un bon candidat pour notre système, principalement en raison du fait qu'il est très robuste aux variations d'images telles que l'éclairage, les transformations affines, et les occlusions partielles d'un objet.

Le SIFT peut être divisé en deux grandes parties : l'extraction et la description de caractéristiques. L'étape d'extraction est basée sur le calcul de la différence de gaussiennes (DoG) et peut être considéré comme un détecteur de blob invariant à l'échelle qui permet de détecter les points «intéressants» dans une image. La deuxième partie considère la région qui entoure le point d'intérêt et constitue un vecteur de description (ou une signature), qui est essentiellement une collection d'histogrammes de gradients normalisés à la rotation.

Après avoir mis en place un prototype logiciel de reconnaissance d'objet spécifique, nous avons remarqué que le goulot d'étranglement qui empêche l'algorithme SIFT de fonctionner en temps réel se trouve dans la partie DoG. Nous avons alors recueilli nos efforts pour migrer cette partie de l'architecture vers du matériel parallélisable d'un FPGA.

La partie DoG nécessite le calcul d'une pyramide d'images, qui est essentiellement une répétition de filtres Gaussien 2D (pour former des échelles), suivi d'un sous-échantillonnage (pour former des octaves). Les résultats des différentes échelles doivent par la suite être synchronisés et soustraits, et les points d'extrêmes locaux à travers les échelles sont considérés comme des points d'intérêt.

Après avoir examiné les implémentations existantes du DoG, nous proposons une nouvelle architecture efficace capable de partager les ressources FPGA bloc RAM pour le filtrage et aussi de réduire la taille des FIFOs de synchronisation. Nous avons également exploité des concepts tels que l'entrelacement d'octaves, la séparabilité des filtres 2D, et la symétrie des filtres. Notre architecture utilise moins de ressources que d'autres implémentations dans la littérature, tout en étant comparable en terme de précision à une implémentation logicielle à virgule flottante. L'implémentation FPGA est capable de rouler à deux ordres de grandeur plus vite qu'une implémentation logicielle équivalente.

De plus, notre architecture est paramétrable en utilisant les génériques VHDL, et nous démontrons comment notre consommation de ressources est affectée en effectuant des balayages de paramètres pour : le nombre d'échelles, le nombre d'octaves, la largeur des filtres, et la largeur maximale de l'image.

Le système a été validé sur une carte de prototypage Xilinx ML605 en utilisant MATLAB et Xilinx System Generator afin de fonctionner en mode co-simulation avec le matériel dans la boucle. L'interface Ethernet est utilisée pour le transfert d'images.

Afin de calculer les descripteurs de caractéristique (deuxième partie du SIFT), nous avons parfois besoin d'envoyer la pyramide d'images à l'hôte, ou de la sauvegarder dans la mémoire externe pour un traitement ultérieur. Après avoir remarqué la similitude des données (dans les sens x et y) et aussi à travers les échelles adjacentes à cause du filtrage, nous sommes arrivés avec une architecture pour un encodeur de Huffman parallèle qui est capable de traiter plusieurs pixels à la fois. Cet encodeur peut être utilisé pour la compression sans perte des images ou des pyramides d'images. Pour les pyramides d'images, nous avons utilisé une version modifiée du prédicteur Paeth qui tient compte de l'échelle adjacente pour l'étape de différenciation et nous avons réussi à atteindre des taux de compression de $27,3\%$ sur la base d'images Caltech-256.

Une autre voie qui a été explorée concerne l'estimation de la carte phosphène. Dans la littérature, certains avaient suggérer une installation de calibration qui consiste à utiliser un écran tactile et des lunettes de réalité virtuelle pour simuler le champ de vue du patient. Le patient est ensuite présenté avec des paires de phosphènes et devra fournir de la rétroaction en utilisant deux doigts pour toucher l'écran tactile et essayer d'évoquer aussi précisément que possible ce qu'il perçoit (distance et angle).

Nous proposons une configuration alternative qui n'utilise pas un écran tactile, mais plutôt une installation d'une caméra qui localise et suit plusieurs marqueurs fiduciaires qui peuvent être manipulés simultanément par le patient pour fournir une rétroaction pour la calibration de la carte phosphène. La caméra sera placée sous une table de surface transparente et les marqueurs fiduciaires seront donc toujours visibles par la caméra. L'image est rectifiée pour enlever les distorsions et les marqueurs sont suivis simultanément. Ceci nous donne l'avantage d'avoir une résolution plus élevé qu'un écran tactile, la possibilité d'utiliser des constellations de phosphènes (au lieu de seulement deux paires à la fois) et la possibilité de modifier l'estimation de la carte calibrée en temps réel en utilisant la rétroaction du patient. Si le nombre de marqueurs est le même que le nombre de phosphènes, l'emplacement physique final des marqueurs sur la table représente directement la carte phosphène qui est effectivement perçue par le patient.

Pour résumer, nous avons identifié l'algorithme SIFT comme un bon candidat pour la reconnaissance d'objets et nous l'avons accéléré en utilisant une nouvelle architecture pour calculer l'extraction de caractéristiques (partie DoG) dans le matériel qui est paramétrable et efficace en termes de consommation de ressources, tout en conservant une précision comparable à une implémentation logicielle. En outre, nous avons identifié la redondance des données dans les images et les résultats intermédiaires pyramidales et nous avons proposé une architecture pour un encodeur de Huffman parallèle qui peut être utilisée pour soulager les goulots d'étranglement. Aussi, nous avons proposé une installation alternative à explorer

pour la calibration et l'estimation de la carte phosphène.

Notre système permet un traitement d'information haut niveau pour l'implant intra-cortical. La compréhension de l'entourage par des méthodes telles que la reconnaissance d'objets nous permettra de simplifier les données avant de les envoyer vers la stimulation sur un nombre limité de phosphènes.

ABSTRACT

Visual impairment may be caused by various factors varying from trauma, birth-defects, and diseases. Until today there are no viable medical treatments for this condition ; hence bio-medical approaches are being employed to overcome that. The Cortivision team has been working on an intra-cortical implant that can bypass the retina and optic nerve and directly stimulate the visual cortex. In this work we aimed to implement a modular, reusable, and parameterizable object recognition system that tends to "simplify" video data prior to stimulation ; hence opening new horizons for partial vision restoration, navigational and even recognition abilities.

We identified the Scale Invariant Feature Transform (SIFT) algorithm as being a robust candidate for our application's needs. A multithreaded software prototype of the SIFT and Lucas-Kanade tracker was implemented to ensure proper overall operation. The feature extractor, difference of Gaussians (DoG) part of the SIFT, being the most computationally expensive, was migrated to an FPGA implementation due to the real-time restrictions that is not achievable on a host machine. The VHDL implementation is highly parameterizable for different application needs and tradeoffs. We introduced a novel architecture employing the sub-kernel trick to reduce resource usage compared to preexisting architectures while still being comparably accurate to a software floating point implementation. In order to alleviate transmission bottlenecks, the system also includes a new parallel Huffman encoder design that is capable of performing lossless compression of both images and scale space image pyramids taking into account spatial and scale data correlations during the predictor phase. The encoder was able to achieve compression ratios of 27.3% on the Caltech-256 data-set. Furthermore, a new camera and fiducial markers setup based on image processing was proposed in order to target the phosphene map estimation problem which affects the quality of the final stimulation that is perceived by the patient.

CONDENSÉ EN FRANÇAIS

Introduction et objectifs

La déficience visuelle, qui est définie par la perte totale ou partielle de la vision, n'est actuellement pas médicalement traitable. Des approches biomédicales modernes sont utilisées pour stimuler électriquement la vision ; ces approches peuvent être divisées en trois groupes principaux : le premier ciblant les implants rétiniens Humayun et al. (2003), Kim et al. (2004), Chow et al. (2004) ; Palanker et al. (2005), Toledo et al. (2005) ; Yanai et al. (2007), Winter et al. (2007) ; Zrenner et al. (2011), le deuxième ciblant les implants du nerf optique Veraart et al. (2003), Sakaguchi et al. (2009), et le troisième ciblant les implants intra-corticaux Doljanu et Sawan (2007) ; Coulombe et al. (2007) ; Srivastava et al. (2007). L'inconvénient principal des deux premiers groupes, c'est qu'ils ne sont pas suffisamment génériques pour surmonter la majorité des maladies de déficience visuelle, car ils dépendent du fait que le patient doit avoir un nerf optique intact et/ou une rétine partiellement opérationnelle ; ce qui n'est pas le cas pour le troisième groupe.

L'équipe du Laboratoire Polystim Neurotechnologies travaille actuellement sur un implant intra-cortical qui stimule directement le cortex visuel primaire (région V1) ; le nom du projet global est Cortivision. Le système utilise une caméra, un module de traitement d'image, un transmetteur RF (radiofréquence) et un stimulateur implantable. Cette méthode est robuste et générique car elle contourne l'oeil et le nerf optique. Un des défis majeurs est le traitement d'image nécessaire pour «simplifier» les données antérieures à la stimulation, l'extraction de l'information utile en écartant les données superflues. Les pixels qui sont capturés par la caméra n'ont pas de correspondance un-à-un sur le cortex visuel comme dans une image rectangulaire, ils sont plutôt mis en correspondance avec une carte complexe de «phosphènes» Coulombe et al. (2007) ; Srivastava et al. (2007). Les phosphènes sont des points lumineux qui apparaissent dans le champ de vision du patient quand le cerveau est stimulé électriquement. Ces points changent en terme de taille, de luminosité et d'emplacement en fonction de la façon dont la stimulation électrique est effectuée (c'est à dire un changement dans la fréquence, la tension, la durée, etc. ...) et même par le placement physique des électrodes dans le cortex visuel.

Les approches actuelles visent à stimuler des images de phosphènes monochromes à basse résolution. Sachant cela, nous nous attendons plutôt à une vision de faible qualité qui rend des activités comme naviguer, interpréter des objets, ou encore lire, difficile pour le patient. Ceci est principalement dû à la complexité de l'étalonnage de la carte phosphène et sa cor-

respondance, et aussi à la non-trivialité de savoir comment simplifier les données à partir des images qui viennent de la camera de façon qu'on conserve seulement les données pertinentes. La Figure 1.1 est un exemple qui démontre la non-trivialité de transformer une image grise en stimulation phosphène.

Des techniques de traitement d'images comme la détection de bords, la segmentation et des approches de vision stéréo ont été proposées dans la littérature comme un moyen de réduire l'information avant la stimulation.

Le concept de réalité augmentée a récemment été introduit. Il s'agit d'un domaine de recherche en informatique qui traite le mélange des données réelles avec des données générées par ordinateur en temps réel.

Un problème principal qui est souvent ignoré, c'est que la carte phosphène n'est pas assez précise ; d'où l'utilisation d'un système en temps réel de reconnaissance d'objets peut amplement contribuer à la simplification de l'environnement pour que les patients non-voyants puissent mieux naviguer dans leur environnement et se doter de capacités de reconnaissance. Les systèmes de déficience visuelle précédents utilisent des techniques de traitement d'images simples qui n'incorporent pas la reconnaissance.

L'objectif serait d'améliorer la qualité ou la méthodologie utilisée pour la perception en temps réel en utilisant des techniques de traitement d'images accélérées sur le matériel parallélisable (FPGA), offrant ainsi une réalité augmentée dans lequel des objets spécifiques peuvent être localisés et/ou reconnus, puis transmis au patient.

Le système doit aussi être paramétrable et indépendant de la plateforme, pour être efficace et réutilisable pour des taches différentes. En outre, la mise en œuvre d'un prototype se servant des ressources FPGA doit être économe en ressources au cas où elle doit être mappée sur des circuits intégrés dédiés.

Les objectifs peuvent être résumés de la façon suivante : 1) Identifier et développer un algorithme logiciel robuste et parallélisable qui peut effectuer la localisation et la reconnaissance d'un nombre limité d'objets sur lesquels il a été formé ; ceux-ci devraient varier dans la pose et l'éclairage. 2) Accélérer les goulots d'étranglement de l'algorithme en utilisant des modules sur matériel (FPGA) qui sont en mesure d'atteindre des performances temps réel. Ces modules doivent être suffisamment génériques pour être réutilisables (peuvent être facilement transférées à une autre plate-forme) et paramétrable, tout en étant aussi efficaces que possible à la fois en termes de ressources et d'utilisation de bande passante. 3) Valider la précision des modules matériels en les comparants à une mise en œuvre équivalente en logiciel. 4) Enquêter sur les méthodes possibles d'estimation et d'étalonnage de la carte phosphène.

Figure 1.2 montre une vue d'ensemble du système qu'on propose. Les détails des différentes parties seront expliqués dans les chapitres à venir. Nous allons utiliser le Scale Invariant

Feature Transform (SIFT) pour extraire et décrire des points clés. La première partie du SIFT sera accélérée dans un FPGA en utilisant une nouvelle architecture pour économiser les ressources. Une base de données d'objet en forme de points-clés est utilisé pour les comparer avec ceux des images qui viennent de la camera. La reconnaissance d'objets est faite quand assez de point-clés sont assortis et après, une stimulation appropriée peut être envoyée aux matrices d'électrodes. Un codeur de Huffman est utilisé pour réduire les goulots d'étranglement HW/SW, et on propose aussi une nouvelle configuration pour la calibration de la carte phosphène.

Un exemple visuel qui décrit l'utilisation du SIFT pour la reconnaissance d'objets est montré dans la Figure 2.8

Étapes de recherche

Nous avons commencé par examiner les différentes approches de traitement d'image et des algorithmes qui peuvent être utilisés à des fins de reconnaissance d'objets. Après cela, nous avons étudié comment les FPGA peuvent être utilisés efficacement pour nos tâches spécifiques d'une manière optimisée.

L'une des méthodes de base de traitement d'image utilisées pour déterminer si un objet (ou plutôt un modèle spécifique) se trouve dans une image d'entrée est la mise en correspondance de modèle. Le modèle est généralement plus petit que l'image cible, et le but serait de déterminer la boîte qui englobe l'objet en question. En termes d'entrainement, cette méthode nécessite un modèle exact de l'image ou de l'objet cible. Le score de correspondance peut être calculée en utilisant de différentes méthodes d'où on peut avoir un compromis entre la complexité de calcul et la robustesse, tels que : la différence des carrés, la corrélation et la corrélation normalisée.

L'avantage principal d'une telle approche réside dans la simplicité algorithmique de la méthode, mais dans des situations réelles il existe plusieurs désavantages, comme ceux qui sont indiquées ci-dessous : 1) La dépendance d'illumination : Si la luminosité du modèle est différente de celle de l'objet dans l'image de recherche, il serait difficile de trouver une bonne correspondance. 2) Gradients d'éclairage : Bien que le processus de normalisation puisse être utilisé pour cibler le problème précédent, un éclairage gradient va probablement échouer même avec la normalisation. 3) La complexité de calcul : Comme il est clair d'après les équations de correspondance, le temps de calcul dépend directement de la taille du modèle. 4) Les transformations affines et d'échelles, la rotation et/ou l'inclinaison de l'objet cible dans l'image de recherche causerait cette méthode d'échouer, car elle ne tient pas en compte les transformations affines ou les distorsions. 5) Occlusion/correspondances partielles : Le

modèle ne doit pas être obstrué et doit avoir plus ou moins une correspondance parfaite sur toute la surface du modèle pour une bonne détection. 6) Choix du Seuil : Choisir le seuil approprié pourrait s'avérer un peu difficile quand on essaie d'adapter le système pour des environnements différents.

Les méthodes basées sur les caractéristiques des images pour la détection d'objets ont culminé dans la littérature, telles que celles qu'on trouve dans Lowe (2004), Bay et al. (2008), Mikolajczyk et Schmid (2005), Bosch et al. (2007), Dalal et Triggs (2005). Ces méthodes ont plusieurs applications tel que la détection d'objets Lowe (2004), Lowe (2001), Lowe (1999), Bay et al. (2008), Bay et al. (2006), le mosaïquage d'images Brown et Lowe (2003), la cartographie et localisation simultanée (SLAM) Montemerlo et al. (2002) Castle et al. (2010) et même la classification de scènes et de contexte Lazebnik et al. (2006).

Les méthodes basées sur les caractéristiques utilisent des patchs (ou caractéristiques) prises autour de points «intéressants» (ou points-clés) dans l'image qui seront utilisées pour déterminer des correspondances.

Au lieu d'utiliser une image de référence comme modèle, nous extrayons les caractéristiques intéressantes à partir de l'image modèle d'un objet et nous essayons ensuite de trouver des correspondances de caractéristiques dans une image différente. Une telle approche va supprimer la plupart des limitations qu'on avait mentionnées précédemment.

Ces algorithmes peuvent être divisés en deux grandes étapes : la localisation (ou l'extraction) de point-clés et la description des caractéristiques. À l'étape de localisation des point-clés, nous essayons d'extraire des points saillants et «intéressants» de l'image (pour qu'on puisse plus tard décrire les caractéristiques ou patchs qui entourent ces points). Les points-clés peuvent être considérés comme des arêtes, des coins, ou même des points invariants à l'échelle.

Les coins d'une image sont généralement considérés comme des bonnes caractéristiques car, ceux-ci ont des variations dans toutes les directions et donc sont relativement stables et peuvent être extraits d'une manière robuste.

En termes de définitions mathématiques, un coin peut avoir de différentes caractéristiques et certaines définitions peuvent être plus robustes en termes de rotation, mise à l'échelle, transformations affines et/ou même pour extraire des blobs. Quelques détecteurs de coins que nous avons étudiés sont : Le détecteur de coins Harris, Shi-Tomasi, détecteur de coins morphologique et les détecteurs de coins multi-échelles.

Les détecteurs de coins multi-échelles ont l'avantage d'être en mesure de relocaliser le même point-clé à une échelle différente ; ce qui est pas mal fréquent dans des situations réelles.

Le résultat des détecteurs de coins, utilisés indépendamment comme coordonnées, peut

être assez bon pour correspondre des caractéristiques ou des points-clés à travers des images avec des mouvements mineurs, par exemple pour des fins de suivre un objet. Mais pour l'exigence d'apparier les caractéristiques ou même des objets entiers à travers des images avec des différences en illumination, échelle et/ou rotation, un détecteur de coin tout seul ne fonctionnera pas. Par conséquence on se fie plutôt à des approches basées sur les caractéristiques tel le «Scale Invariant Feature Transform (SIFT)» Lowe (2004) ; Ces approches rajoutent une description ou une signature aux coins (ou point-clés) afin d'obtenir des caractéristiques distinctives et discriminantes.

Nous ferons appel à l'algorithme SIFT à cause de sa robustesse et sa capacité de faire correspondre des caractéristiques avec des variations d'échelle, des changements de point de vue, des rotations, des différences d'éclairage et même des occlusions partielles. De plus, les caractéristiques sont assez distinctives pour les comparer à une grande base de données, ce qui rend le SIFT idéal pour des applications telles que la détection d'objets, la classification de scènes et la mise en correspondance d'images.

La méthode peut être divisée en deux parties majeures :

- L'extraction de point-clés : Cette partie est comparable à une extraction coin multi-échelle (ou l'extraction blob) et est basé sur la pyramide d'images et plus spécifiquement la différence de Gaussiennes (Difference of Gaussians DoG).
- Description de points-clés : Les patches autour d'un point clé sont choisies et caractérisées pour être utilisées comme un vecteur de signature en utilisant des histogrammes de gradients qui sont normalisés en angle de rotation.

Après avoir identifié un algorithme approprié pour notre application nous avons étudié les différentes techniques de traitement d'images à utiliser sur FPGA. Nous avons défini des interfaces qui peuvent être montées en cascade et qui respectent le contrôle de flux simple. En raison que les données sont en forme de flux de pixels et le fait que les FPGA sont généralement limités en termes de mémoire interne, nous pouvons diviser les opérateurs de traitement d'images en matériel en trois grandes catégories : 1) Traitement par pixel (aucune mémoire tampon nécessaire) 2) Traitement par voisinage (des blocs RAM internes peuvent être utilisés) 3) Traitement par trame d'image (une RAM externe sera nécessaire)

Nous avons également étudié les différentes architectures pour les filtres à réponse impulsionnelle finie qui optimisent les ressources et l'efficacité en terme de fréquence et latence.

Nous avons commencé par faire un prototype logiciel du système en question afin d'évaluer la faisabilité et la robustesse. Nous avons utilisé la librairie OpenCV de traitement d'images à cet effet. L'algorithme SIFT était lent pour des performances temps réel et donc nous l'avons combiné avec l'algorithme Lucas-Kanade de flux optique pour suivre un objet qui a été reconnu. Le principal goulot d'étranglement de l'algorithme SIFT est la partie de différence

de Gaussiennes (DoG) dont nous avions l'intention d'accélérer sur le FPGA.

En ce qui concerne la mise en œuvre du matériel, nous avons évalué de différentes méthodes de mise en œuvre de la différence de gaussiennes et nous avons réussi à implémenter une nouvelle architecture qui utilise une technique de partage de bloc de mémoire RAM pour les opérations de voisinage ainsi que l'entrelacement des données, la séparation des filtres 2D, et l'exploitation de la symétrie du filtre afin de réduire l'utilisation des ressources par rapport à d'autres architectures dans la littérature tout en maintenant la précision.

Notre architecture est paramétrable en utilisant des génériques VHDL en termes d'échelles, des octaves, de nombres de coefficients de filtres et de largeur maximale des images.

La plateforme de prototypage Xilinx ML605 a été utilisée en conjonction avec MATLAB et Xilinx System Generator. Nous avons testé notre implémentation en utilisant de la co-simulation matérielle, c'est à dire en utilisant le réseau Ethernet pour effectuer les I/O et en testant avec le matériel en boucle.

Dans de nombreux cas, la pyramide d'image ou les images doivent être transférées à l'hôte (ou mémoire externe) pour le traitement de niveau supérieur. Nous avons remarqué deux choses, tout d'abord toutes les images floues (après le filtrage Gaussien) ont tendance à avoir très peu de composantes haute fréquence dans les directions x et y. Donc l'histogramme d'une image de différence spatiale et causale aurait tendance à avoir un pic très raide autour de zéro. Ces images sont bien compressibles à l'aide d'un encodeur de Huffman.

Nous avons donc proposé une architecture parallèle pour un encodeur de Huffman qui est capable de traiter plusieurs pixels en parallèle et de manière adéquate peut être utilisée pour compresser des images individuelles ainsi que des pyramides d'image complètes. Pour la compression de pyramides d'images, un prédicteur de Paeth modifié a été utilisé. Une telle méthode de compression permettrait d'atténuer les goulots d'étranglement de bande passante sans taxer l'exactitude en raison de sa nature de compression sans perte.

Une autre voie qui a été explorée concerne l'estimation de la carte phosphène. En générale, après avoir effectué la reconnaissance des objets, une représentation de cette information devra être transmise au patient. La représentation peut être évoquée par des signaux audibles, ou même comme des stimulations intra-corticales pour le cas du projet Cortivision.

Pour qu'un tel système fonctionne, on dépend sur une supposition majeure que la carte phosphène est déjà connue. Une carte approximative du cerveau où le placement physique des électrodes se traduirait par un phosphène étant éclairé spécifiquement dans le champ de vision a été présenté dans Dobelle et Mladejovsky (1974). Une telle carte est généralement plus dense vers le centre du champ de vision et il ne faut pas s'attendre à obtenir une grille parfaitement uniforme (comme une image de pixels).

Typiquement la carte phosphène peut être calculée avec l'aide du patient et ses réactions

afin de minimiser l'erreur en estimant la position des phosphènes. Notez que si la carte est déformée, la stimulation sera également déformée ce qui compliquera la reconnaissance.

RÉSULTATS

Une implémentation paramétrable en VHDL de la différence de gaussiennes et encodeur de Huffman a été mise en œuvre sur un FPGA Xilinx Virtex-6 ML605 et testée en co-simulation de matériel dans «System Generator». L'architecture peut fonctionner à 100 MHz, émet un résultat à tous les deux cycles d'horloge (en raison de l'architecture entrelacée) et est à deux ordres de grandeur plus rapide qu'une mise en œuvre en logiciel Vedaldi et Fulkerson (2010) pour des images de 640×480 testées sur une machine hôte (Intel Core i5@2.27 GHz, 6 Go de RAM).

En termes de précision numérique, nous avons montré que l'architecture utilisée est comparable à un modèle de logiciel en double précision utilisant des filtres de 31-coefficients. Notez que nous pouvons réduire davantage la consommation de ressources en réduisant la précision.

L'ensemble de données d'images Caltech-256 a été utilisé pour tester les taux de compression de l'encodeur Huffman. Une table fixe Huffman a été construise en prenant 10 images d'entrainement de chaque catégorie d'objets. La même table a été utilisée pour encoder toutes les images et leurs pyramides qui sont constitués de combinaisons octaves/échelles. Les résultats de compression sans perte de $27,3\%$ ont été réalisés sur l'ensemble de données d'images Caltech-256.

Conclusion

Notre objectif principal était de traiter les images afin de «comprendre» le contenu des images et être capable de simplifier les données par le biais d'un système temps réel embarqué de reconnaissance d'objets.

Nous avons principalement contribué en introduisant une nouvelle architecture pour la différence de gaussiennes (DoG) qui partage les blocs RAM pour les opérations de voisinage afin d'utiliser les ressources FPGA efficacement tout en conservant la précision et en augmentant la vitesse de traitement de deux ordres de grandeur par rapport à une implémentation logicielle. Nous avons également fourni une méthodologie d'utiliser un prédicteur Paeth modifié combiné avec une architecture Huffman du codeur parallèle pour compresser les pyramides d'images ainsi que des images courantes permettant d'économiser de la bande passante de transmission. Par ailleurs, nous avons proposé une alternative qui peut en outre être explorée pour l'estimation carte phosphène basée sur le suivi des marqueurs repères en

utilisant des outils de traitement d'image.

Nous avons adopté l'algorithme SIFT pour la reconnaissance d'objets. L'algorithme s'est avéré être assez robuste pour gérer les variations d'image différents tels que la luminosité, le contraste, redimensionnement, de rotation et de transformations affines.

Un prototype logiciel a été implémenté comme une application multithread qui fusionne les algorithmes SIFT et Lucas-Kanade pour la reconnaissance et la poursuite d'objets. Le SIFT n'est pas capable de fonctionner en temps réel pour le traitement vidéo sur un ordinateur hôte, car il nécessite environ une seconde pour le traitement d'une seule image et à détecter un objet. Le suiveur a été utilisé pour surmonter cette limitation de suivi de l'objet reconnu en temps réel.

Après avoir identifié la différence de gaussiennes comme étant le goulot d'étranglement de l'algorithme SIFT, nous sommes passés à la migration de cette partie de la mise en œuvre d'un FPGA.

Nous nous sommes concentrés sur le codage d'une architecture hautement paramétrable qui sera en mesure de cibler les besoins des différentes applications, soit la précision par rapport aux arbitrages d'utilisation de ressources et le codage fixe de l'un des paramètres. En outre, le VHDL a été codé pour être réutilisable et portable.

En reconnaissant un traitement de flux de données, nous avons examiné les différentes techniques de traitement d'images pour les implémentations FPGA.

Afin de parvenir à une conception efficace des ressources, la mise en œuvre de notre DoG utilise des concepts tels que : pipeline, les données d'octave entrelacement, la séparabilité du filtre, la symétrie du filtre et le partage de blocs RAM pour les opérations de voisinage que nous avons introduit.

Des balayages de paramètres (largeur, nombres de coefficients de filtres, octaves, échelles) de l'architecture ont été réalisés et tracées et aussi comparés par rapport à d'autres architectures en termes d'estimation des ressources FPGA.

La mise en œuvre DoG matériel a également été démontrée d'être comparable en terme de précision à une mise en œuvre logiciel en virgule flottante.

Par ailleurs, nous avons introduit une nouvelle architecture pour un encodeur Huffman parallèle qui peut être utilisée à la fois pour pyramide d'images ainsi que la compression d'image simple. Nous avons obtenu des taux de compression sans perte de $27,3\%$ sur l'ensemble de données d'images Caltech-256.

Nous avons également proposé une nouvelle méthodologie d'effectuer la calibration de carte de phosphènes en utilisant une installation simple qui emploie une caméra filmant une table avec des marqueurs de repère qui peuvent être manipulés par le patient qui fournira une sorte de rétroaction directe.

Recommandations

En raison de l'entrelacement d'octaves de données, le module de DoG est capable de traiter 1 pixel à chaque 2 cycles d'horloge. Cette limitation peut être surmontée par le traitement de plusieurs pixels à la fois, quelques-uns des modules qui sont mis en œuvre supportent déjà cette fonctionnalité.

Une autre limitation réside dans les tables de Huffman. Idéalement, le codeur de Huffman devrait être en mesure d'utiliser des tables adaptatives.

Une autre amélioration sera la possibilité de calculer la deuxième partie de l'algorithme SIFT (les vecteurs de description) sur un processeur embarqué comme ARM qui est présent dans la nouvelle famille Xilinx Zynq.

Les caractéristiques SIFT peuvent également être utilisées pour la reconnaissance de scène comme indiqué dans Lazebnik et al. (2006). Cette information peut être intégrée dans le système pour modifier la base de données qui est utilisée pour la recherche d'objet en fonction du contexte, par exemple si le patient est dans une cuisine, le système interrogera une base de données d'articles de ménage, alors que s'il est à l'extérieur, une base de données de voitures, motos et de panneaux d'arrêts pourrait être utilisée. Cela permettra de réduire le nombre de faux positifs et de comparaisons inutiles.

Enfin, la technique de cartographie phosphène peut être testée avec la rétroaction de patients pour intégrer un ensemble de modèles par exemple en utilisant des formes géométriques de base et des objets mouvants. Un émulateur peut être configuré avec des lunettes de réalité virtuelle pour modéliser le champ de vision du patient et le patient pourra manipuler les marqueurs de repères pour calibrer la carte phosphène perçue en temps réel.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

RÉSUMÉ iv

ABSTRACT viii

CONDENSÉ EN FRANÇAIS ix

TABLE OF CONTENTS xviii

LIST OF TABLES xxi

LIST OF FIGURES xxii

LIST OF ACRONYMS AND ABBREVIATIONS xxiv

CHAPTER 1 INTRODUCTION 1
1.1 Theoretical Framework 1
1.2 Research Question 3
1.3 Hypothesis 3
1.4 Research Goals 3
1.5 Objectives 4

CHAPTER 2 LITERATURE REVIEW 6
2.1 Template Matching 6
2.1.1 Border Handling 7
2.1.2 Matching Scores 7
2.1.3 Advantages/Disadvantages 8
2.2 Feature-Based Methods 10
2.2.1 Corner Detectors 12
2.2.2 Scale Invariant Feature Transform 15
2.2.3 Other Feature-Based Algorithms 21
2.3 Image Processing Techniques in FPGAs 22
2.3.1 Image Protocols and Interfaces 23
2.3.2 Types of FPGA Image Processing Operators 23

2.3.3 Finite Impulse Response 30

CHAPTER 3 APPROACH AND ORGANIZATION 34
3.1 Software Prototype 34
3.1.1 Image Processing Libraries 34
3.1.2 SIFT 34
3.1.3 Keypoint Matching 34
3.1.4 Object Localization 35
3.1.5 Lucas-Kanade 36
3.2 Hardware Implementation 37
3.2.1 Xilinx System Generator 37
3.2.2 VHDL Coding 40
3.2.3 Architecture Optimization 41
3.2.4 Huffman Encoding 44
3.3 Phosphene Map Estimation 46

CHAPTER 4 AN IMAGE PROCESSING SYSTEM DEDICATED TO A VISUAL INTRA-CORTICAL STIMULATOR 48
4.1 Introduction 48
4.2 Difference of Gaussians 52
4.3 Parallel Huffman Encoder 55
4.3.1 Image Differentiation 56
4.3.2 Architecture of the Proposed Encoder 57
4.4 Phosphene Map Calibration 58
4.5 Simulation and Experimental Results 61
4.5.1 DoG Precision 61
4.5.2 VHDL Synthesis 62
4.5.3 Huffman Encoding 64
4.6 Conclusion 65

CHAPTER 5 GENERAL DISCUSSION 67
5.1 Algorithm Robustness 67
5.2 Hardware Implementation 67

CHAPTER 6 CONCLUSION AND RECOMMENDATIONS 70
6.1 Research Synthesis 70
6.2 Future Recommendations 72

6.3 Concluding Remarks 72

REFERENCES 74

LIST OF TABLES

Tableau 4.1 Precision of cascaded DoG and sub-kernel DoG 63
Tableau 4.2 DoG Resource Usage 63
Tableau 4.3 Parallel Huffman encoder resource usage 64
Tableau 4.4 Huffman Encoder Compression Ratio 66

LIST OF FIGURES

Figure 1.1 Example of mapping a gray image onto the phosphene map 2
Figure 1.2 Overview of the proposed object recognition approach 4
Figure 2.1 Template matching : Template (a) is swept over the search image (b) until a match is found. 6
Figure 2.2 Distortions that will cause template matching to fail. 9
Figure 2.3 Feature-based object recognition. 10
Figure 2.4 Feature matching - Good features (green) have stable locations and are distinctive. Bad features (red) can be easily mistaken for one another. . 11
Figure 2.5 Edge features are bad since they tend to slide along the edge and are similar in nature. 12
Figure 2.6 Morphological operators applied to a binarized image. 5×5 square structuring element were used in this case. 14
Figure 2.7 Morphological kernels used for corner detection : Cross ; Diamond ; X ; Box. 15
Figure 2.8 SIFT example : Object recognition 17
Figure 2.9 Image pyramid. 18
Figure 2.10 Gaussian Scale Space (GSS). 18
Figure 2.11 Difference of Gaussians (DoG). 19
Figure 2.12 SIFT descriptor. Note : The figure shows a sample 2×2 grid of 8 bin histograms, whereas the actual algorithm uses a 4×4 grid. 21
Figure 2.13 Module with a generic FIFO interface. 23
Figure 2.14 RGB to YUV output 25
Figure 2.15 Flat Field Correction - Pixel-based. 26
Figure 2.16 [5×5] Kernelizer. Line buffers are used to form a moving window around the required pixel 27
Figure 2.17 SRL16E - LUT can be configured as a shift register (maximum 16 latency). 27
Figure 2.18 FPGA external RAM buffering. 28
Figure 2.19 Multi-camera synchronization 28
Figure 2.20 Image rotation - Backward mapping 29
Figure 2.21 FIR - Direct form using an adder tree 31
Figure 2.22 FIR - Transpose form 31
Figure 2.23 FIR - Systolic form 32

Figure 2.24 FIR - Multiply Accumulate (MACC) . . . 32
Figure 3.1 Snapshot of the multithreaded SIFT and Lucas-Kanade tracker. . . . 37
Figure 3.2 Xilinx System Generator - Basic inverter. . . . 38
Figure 3.3 Hardware co-simulation test bench. . . . 39
Figure 3.4 $[7 \times 7]$ Kernel. $[5 \times 5]$ and $[3 \times 3]$ Sub-kernels. . . . 42
Figure 3.5 Cascaded approach for 2 scales. . . . 42
Figure 3.6 Sub-kernel approach for 2 scales. . . . 43
Figure 3.7 Gaussian blurring and differentiation (Diff.) effect on the image histogram. . . . 45
Figure 3.8 Example usage of Huffman encoder. . . . 46
Figure 3.9 Original arrow symbol (left), distorted symbol (right) due to errors in phosphene map estimation. . . . 47
Figure 4.1 Overview of the Cortivision system. . . . 49
Figure 4.2 SIFT : Image Pyramid. 2 Octaves 4 Scales. Blurring at every scale and subsampling (2×2) at every octave. Notation : Gaussian Scale Space (GSS) and Difference of Gaussians (DoG). . . . 50
Figure 4.3 Image processing system used for object recognition and tracking . . . 51
Figure 4.4 Interleaved DoG. (3 Octaves, 4 Scales) Notation : Kernelizer (Ker.) and Interleaved (Inter.) . . . 52
Figure 4.5 2D Interleaved FIR filter. Taking advantage of the of 2D Gaussian separability, we save resources by performing a 1D vertical filtering followed by a 1D horizontal one. . . . 54
Figure 4.6 Symmetric FIR with pre-adder. Notation : Registers are indicated with a triangular notch. . . . 55
Figure 4.7 Simplified Block Diagram of Parallel Huffman Encoder . . . 57
Figure 4.8 Barrel Shifter (a) Basic Architecture (b) Multiplier-based *(Notation : $\gg$ logical bit shift)* . . . 58
Figure 4.9 Parallel Huffman Barrel Shift Register . . . 59
Figure 4.10 Set-up of the phosphene map calibration. . . . 60
Figure 4.11 DoG absolute mean error, compared to a double precision 31-tap kernel (for all octave/scale combinations). . . . 61
Figure 4.12 Cascaded DoG Resource Usage – Parametric Sweep . . . 62
Figure 4.13 Sub-Kernel DoG Resource Usage – Parametric Sweep . . . 64
Figure 4.14 Compression Buffering Scheme . . . 65
Figure 4.15 Caltech-256 samples : mars, galaxy, mussels & trilobite . . . 65
Figure 5.1 BRAM utilization estimate . . . 68

LIST OF ACRONYMS AND ABBREVIATIONS

BBF	Best-Bin-First
BPP	Bits per Pixel
BRAM	Block Random Access Memory
BW	Bandwidth
CORDIC	Coordinate Rotation Digital Computer
DDR SDRAM	Double Data Rate Synchronous Dynamic Random Access Memory
DEMUX	Demultiplexer
DoG	Difference of Gaussians
DSP	Digital Signal Processing
FFC	Flat Field Correction
FFT	Fast Fourier Transform
FIFO	First In First Out
FIR	Finite Impulse Response
FOV	Field of View
FPGA	Field Programmable Gate Array
FPS	Frames per Second
GSS	Gaussian Scale Space
HSV	Hue Saturation Value
HW	Hardware
LK	Lucas-Kanade
LoG	Laplacian of Gaussians
LUT	Look-Up Table
MACC	Multiply Accumulate
MUX	Multiplexer
PCA	Principal Component Analysis
RAM	Random Access Memory
RANSAC	Random Sample Consensus
RGB	Red Green Blue
RLDRAM	Reduced Latency Dynamic Random Access Memory
RLE	Run Length Encoding
ROI	Region of Interest
ROM	Read Only Memory
SAD	Sum of Absolute Differences

SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SRAM	Static Random Access Memory
SURF	Speeded Up Robust Features
SW	Software
VHDL	VHSIC hardware description language
VHSIC	Very-High-Speed Integrated Circuits

CHAPTER 1

INTRODUCTION

Visual impairment, defined by the full or partial loss of vision, as of today is not medically treatable. The causes of such a handicap vary widely and can range anywhere between trauma and congenital issues.

Modern bio-medical approaches are being employed to recover or stimulate vision electrically ; these may be divided into three main groups : the first one targeting retinal implants Humayun *et al.* (2003); Kim *et al.* (2004); Chow *et al.* (2004); Palanker *et al.* (2005); Toledo *et al.* (2005); Yanai *et al.* (2007); Winter *et al.* (2007); Zrenner *et al.* (2011), the second optic nerve implants Veraart *et al.* (2003); Sakaguchi *et al.* (2009), and the third ones involving cortical implants Doljanu and Sawan (2007); Coulombe *et al.* (2007); Srivastava *et al.* (2007). The main drawback of the former two is that they are not generic enough to cope with most of the visual impairment diseases, since they rely on the patient having an intact optic nerve and/or a partially operational retina. This is not the case for the third. Other non-invasive methods that may involve stimulation by sound are also being used, but these are not well agreed upon to fit in the same scope.

1.1 Theoretical Framework

The *Polystim Neurotechnologies* team is currently working on an intra-cortical implant that directly stimulates the primary visual cortex (V1 region) ; the global project name is *Cortivision*. The process uses a front-end camera, an image processing module, an RF (Radio Frequency) link, and an implantable stimulator for the purpose ; this method is robust and generic since it bypasses the eye and optic nerve. One of the major challenges is the image processing required to "simplify" the data prior to stimulation, extracting all of the useful information and discarding the superfluous data. The pixels that are being captured by the camera do not have a one-to-one mapping on to the visual cortex as in a rectangular image ; they are rather mapped into a complicated map of "phosphenes" Coulombe *et al.* (2007); Srivastava *et al.* (2007) which are bright dots that are perceived by the patients when the brain is electrically stimulated. These dots change in size, illumination, and location depending on how the electrical stimulation is performed (i.e. a change in frequency, voltage, duration, etc. . .) and even by the electrode placement in the cortex.

Current approaches aim at stimulating low-resolution grayscale phosphene images. Knowing

that, we would expect a rather low-quality vision making it hard for the patient to navigate, interpret objects, or even read. This is mainly due to the complexity of the phosphene mapping and also to the non-triviality of knowing how to simplify the data in the captured images in such a way that only relevant data is kept.

Figure 1.1 is an example that shows the complexity of illustrating a captured input video stream onto a pre-calibrated phosphine map. The input image is shown on the left, the middle image shows the available phosphene map (this is what the subject would be "seeing" in case we showing a stimulating a white image or a white wall) , and the leftmost image shows the mapping of the gray image onto the available phosphenes using the local gray value.

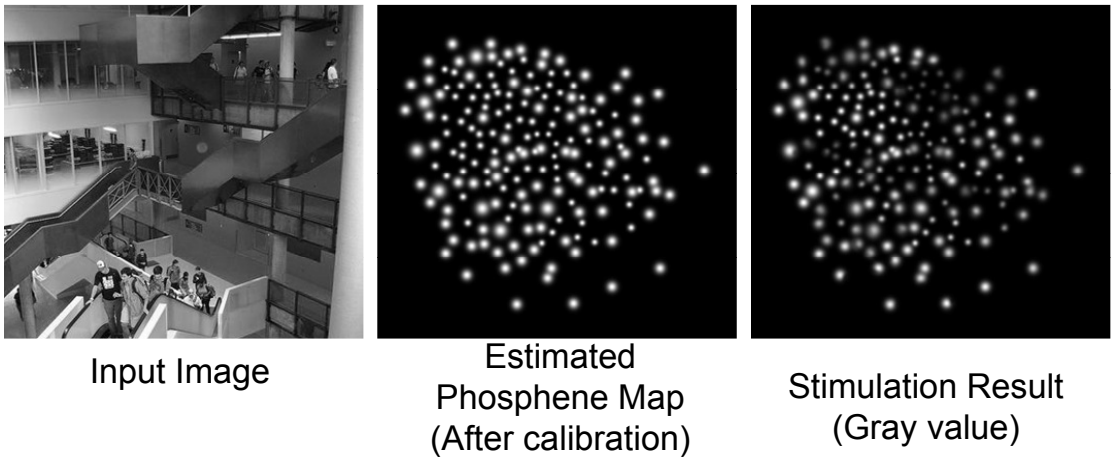


Figure 1.1 Example of mapping a gray image onto the phosphene map

Edge detectors, segmentation, and stereo vision approaches have been proposed in the literature as a way of reducing information prior to stimulation. A stereo vision module has been implemented in Cortivision ; the output of such a module is a disparity map in which close surfaces are indicated by a brighter intensity (or gray value) than ones that are further away (black). The main drawbacks of this module is that it uses an infrared-light pattern projector and a camera to achieve its purpose, which may be cumbersome, and the actual output may not be meaningful enough by itself for the patient to interpret, which is why we plan to add a degree of processing upon that.

The concept of augmented reality has recently been introduced and is proving to be quite appealing. It is a field of computer research which deals with blending real-world data with computer-generated data in real-time ; a common example is using special projective eyeglasses to overlay text on the real-world (this could indicate shopping malls, object dimensions, etc. . .). Some medical applications intended for visual impairment Toledo *et al.* (2005) have relied on that concept being implemented on Field Programmable Gate Arrays

(FPGAs) ; such approaches are used in retinal implants where the patient still has some partial vision.
We would be investigating the use of such a system in terms of high-speed image processing modules and techniques that can be hardware implementable on FPGAs and or a mix of hardware/software processing to attain real-time performance. Such techniques would be targeted to provide a simplified and interactive environment for the patient.

1.2 Research Question

How to come up with a parameterizable and reusable system that can process, simplify, and present the captured image stream in real-time in order to provide a more useful mapping prior to stimulation ?

1.3 Hypothesis

- Hypothesis : The phosphene map is not accurate enough, hence the use of a real-time object recognition system is required to simplify the environment for visually impaired patients and lead to better navigation and recognition abilities.
- Justification of originality : Augmented reality systems are quite recent in the literature, and are usually overlaid over a person's natural vision system (being handicapped or not), but in this case it overrides the actual vision and is interacting directly with the visual cortex to provide better context awareness. Moreover, previous visual impairment systems rely on simple image processing tasks rather than recognition to convey information to the patient.
- Refutability : The hypothesis would be refuted if the system cannot operate in real-time or is not robust enough to varying conditions.

1.4 Research Goals

The research goals would be to improve the quality or methodology used for perception by means of real-time image processing techniques accelerated on parallelizable hardware (FPGA), hence providing an augmented reality in which specific objects can be localized and/or recognized and then passed on to the patient.

The system should be parameterizable and possibly platform-independent in order to be efficiently reusable. Moreover, the FPGA implementation should be resource-efficient in case it needs to be mapped on to smaller devices.

1.5 Objectives

- Identify and develop a robust and parallelizable software algorithm that can perform localization and recognition on a limited number of objects on which it was trained; these are expected to vary in pose and illumination.
- Accelerate the algorithm's bottlenecks using hardware (FPGA) modules to be able to achieve real-time performance. These modules should be generic enough to be reusable (can be easily transferred to a different platform) and parameterizable while being as efficient as possible on both resource and bandwidth utilization.
- Validate the hardware modules' accuracy as compared to a software equivalent implementation.
- Investigate possible methods for estimating the phosphene map.

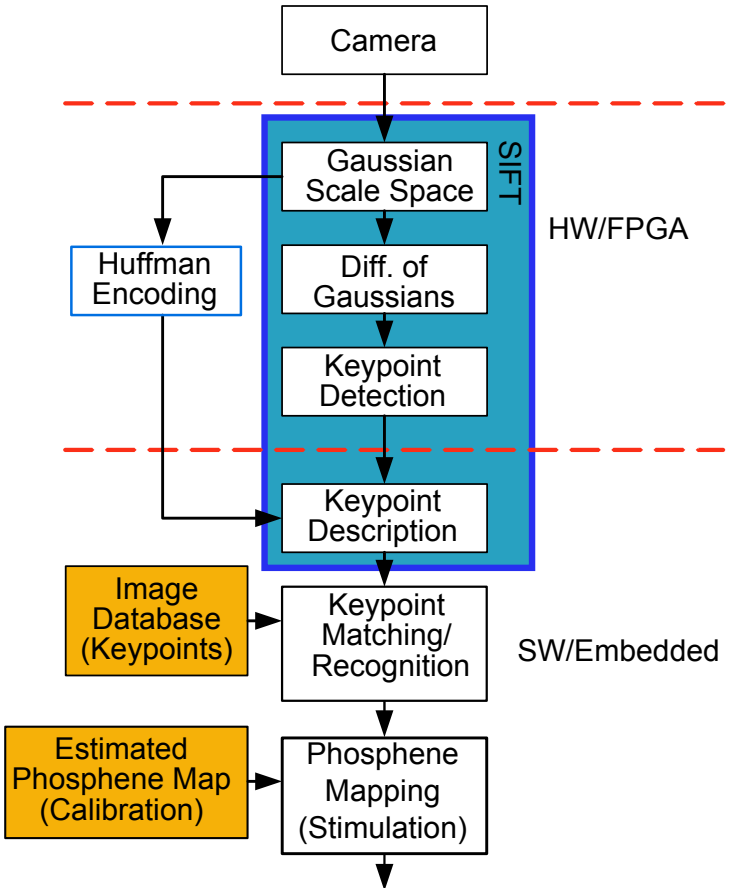


Figure 1.2 Overview of the proposed object recognition approach

Figure 1.2 shows an overview and the intended roadmap that we will be following throughout the text. The details of the different parts will explained in the chapters to come.

Basically we would be using the Scale Invariant Feature Transform (SIFT) to extract and describe keypoints. The first part of SIFT (being computationally expensive) will be accelerated in an FPGA using a novel architecture to save resources. An object database of keypoints is kept and compared to the incoming frames, and whenever enough keypoints are matched, an object is recognized and afterwards an appropriate stimulation image can be sent to the electrode matrices. A Huffman encoder is to be used to reduce HW/SW transmission bottlenecks and also a new setup for calibrating the phosphene map is proposed.

In what follows we will be investigating different image processing techniques that can be used for the purpose of our application. After having identifying a suitable algorithm and validating it in software, we will accelerate the computationally intensive parts using reconfigurable hardware in order to be able to support real-time operation. While doing so, we propose a novel architecture for sharing and saving hardware resources. Moreover, the architecture, as will be seen in throughout the document, will be coded in a highly parameterizable fashion in order to be reusable and portable. An architecture for a hardware Huffman encoder is also proposed that can treat pixels in parallel and can be used for data reduction. Finally, we suggest a viable setup for phosphene map estimation that makes use of fiducial markers and image processing techniques as an alternative to the current tactile screen approaches.

A literature review is covered in Chapter 2, followed by the overall approach that was used in Chapter 3, the journal article entitled “An Image Processing System Dedicated to a Visual Intra-Cortical Stimulator” in Chapter 4, a general discussion in Chapter 5, and we will then conclude and discuss future improvements in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

This chapter provides an overview of image processing approaches and algorithms that can be used for the purpose of object recognition ; Moreover we will also review how FPGAs are used in an image processing context by applying certain design rules and differentiating between the various processing functions.

2.1 Template Matching

Template matching is one of the basic image processing methods used to determine if an object (or rather a specific template) is found in an input image. The template is typically smaller than the target image, and the purpose would be to determine the bounding box of the object in question. In terms of training, this method requires an exact template of the image to be pre-collected such as the one shown in Figure 2.1a and to be matched in a larger image as indicated with the green rectangle in Figure 2.1b.

(a) (b)

Figure 2.1 Template matching : Template (a) is swept over the search image (b) until a match is found.

In essence, template matching works by sliding a template T in a raster scan order (top-left to bottom-right) over the input image I and calculating a correlation factor or matching

score to determine whether we have a positive match.
NOTE : in terms of image processing terminology, correlation and convolution are typically used interchangeably.

In what follows, we will go over the border handling techniques, the various scores of template matching and the advantages and disadvantages of this method.

2.1.1 Border Handling

Assuming that the template and target images have the respective dimensions (Width∗Height) of $M * N$ and $W * H$. Obviously, we will encounter the issue of border errors where the template would actually surpass the image boundary. For that, either the template will only be scanned *inside* the input image and the resulting "match image" would have smaller dimensions than the original, or the input image would have to be border padded, resulting in an input image of dimensions $(W + M - 1) * (H + N - 1)$. Border handling would be performed in one of the following fashions :

- Zero padding : the border would simply be padded with zeros.
- Constant padding : a constant value (e.g. 128) would be used for the borders.
- Border replication : The edge pixel is replicated for the whole border.
- Mirroring : The image borders would be a mirrored version of the input image. This technique tends to minimize the artifacts in convolution operators and will work well for symmetric templates.

NOTE : These border handling techniques are also used in any image processing convolution operations (e.g. Gaussian blurring) as will be seen later on in Section 2.2.2.

2.1.2 Matching Scores

In order to determine a positive match, a certain match score or result image R has to be calculated ; afterwards a threshold is chosen to detect a match. Various ways of calculating a matching score have been proposed in the literature Brunelli (2009),Bradski and Kaehler (2008) ; these can be summarized as follows.

Squared Difference

Assuming that T is the template and I is the search image ; the simplest of the methods would be to calculate the sum of squared differences

$$R(x, y) = \sum_{x'} \sum_{y'} [T(x', y') - I(x + x', y + y')]^2 \tag{2.1}$$

or the sum of absolute differences (SAD)

$$R(x,y) = \sum_{x'} \sum_{y'} |T(x',y') - I(x+x', y+y')| \tag{2.2}$$

a lower match score (closer to zero) would actually indicate a better match.

Correlation

Alternatively, correlation can be used as the match measure

$$R(x,y) = \sum_{x'} \sum_{y'} T(x',y') * I(x+x', y+y') \tag{2.3}$$

A higher score in this case would indicate a better match. It should be noted that this method would prove to be sensitive to white areas in the image which would result in false positives.

Normalized Correlation

By normalizing the previous method, we end up with what is known as the normalized correlation

$$R_n(x,y) = \frac{\sum_{x'} \sum_{y'} T(x',y') * I(x+x', y+y')}{\sqrt{\sum_{x'} \sum_{y'} T(x',y')^2 * \sum_{x'} \sum_{y'} I(x+x', y+y')^2}} \tag{2.4}$$

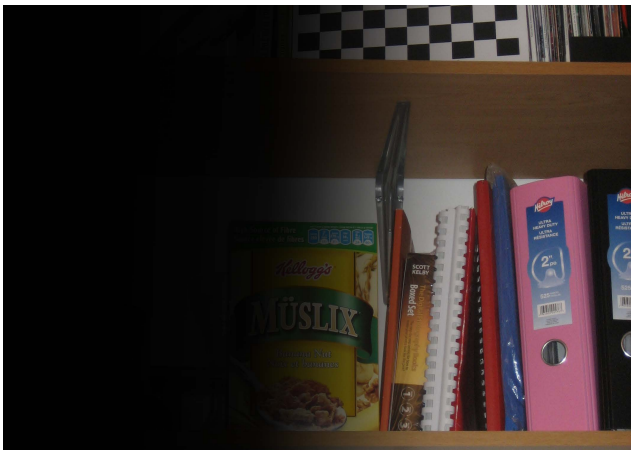
The normalization step alleviates the issues where there is a mismatch in illumination between the template and the input search image and even the false matches that could result from white patches. This improvement definitely comes at the high computational cost.

2.1.3 Advantages/Disadvantages

Template matching is one of the straight forward methods used for object detection. Its main advantage would lie in the algorithmic simplicity of the method, which works a bit as a brute force approach. But, while it might prove to be useful for applications in controlled environments (e.g. machine vision and inspection Crispin and Rankov (2007)), it has many draw backs in real life situations, mainly the ones cited below :

- Illumination dependency : If the template brightness is different from the search image's one, a good match might not be trivial to attain.
- Illumination gradients : While the normalization process can be used to target the previously mentioned problem, a gradient illumination (see Figure 2.2a) would probably fail.

– Calculation complexity : As would be clear from the matching equations, the calculation time is directly dependent on the template's size.
– Affine transformations : Scaling, rotating, and/or skewing the target object in the search image would cause this method to fail, since it does not account in any way for any affine transforms or distortions (see Figure 2.2b).
– Occlusions/partial matches : The template is expected not be occluded and to have more or less a perfect match over the whole area of the template for proper matching.
– Threshold choice : Choosing the appropriate threshold might prove to be a bit tricky when one is trying to accommodate the system for different operating environments.



(a) Artificial Lighting Gradient

(b) Distorted Template

Figure 2.2 Distortions that will cause template matching to fail.

2.2 Feature-Based Methods

Feature-based methods for object detection have peaked in the literature such as the ones found in Lowe (2004), Bay *et al.* (2008), Mikolajczyk and Schmid (2005), Bosch *et al.* (2007), Dalal and Triggs (2005). Applications for such methods include object detection Lowe (2004), Lowe (2001), Lowe (1999), Bay *et al.* (2008), Bay *et al.* (2006), image mosaicing Brown and Lowe (2003), Simultaneous Mapping and Localization (SLAM) Montemerlo *et al.* (2002) Castle *et al.* (2010) and even scene classification Lazebnik *et al.* (2006). Note that we will be using the terms feature and keypoint interchangeably to describe an interesting point in the image.

Feature-based methods differ from basic template matching ones in that a subset of patches (or features) taken around "interesting" points (or keypoints) in the image would be used to determine matches. A caricaturized description of these approaches is shown in Figure 2.3.

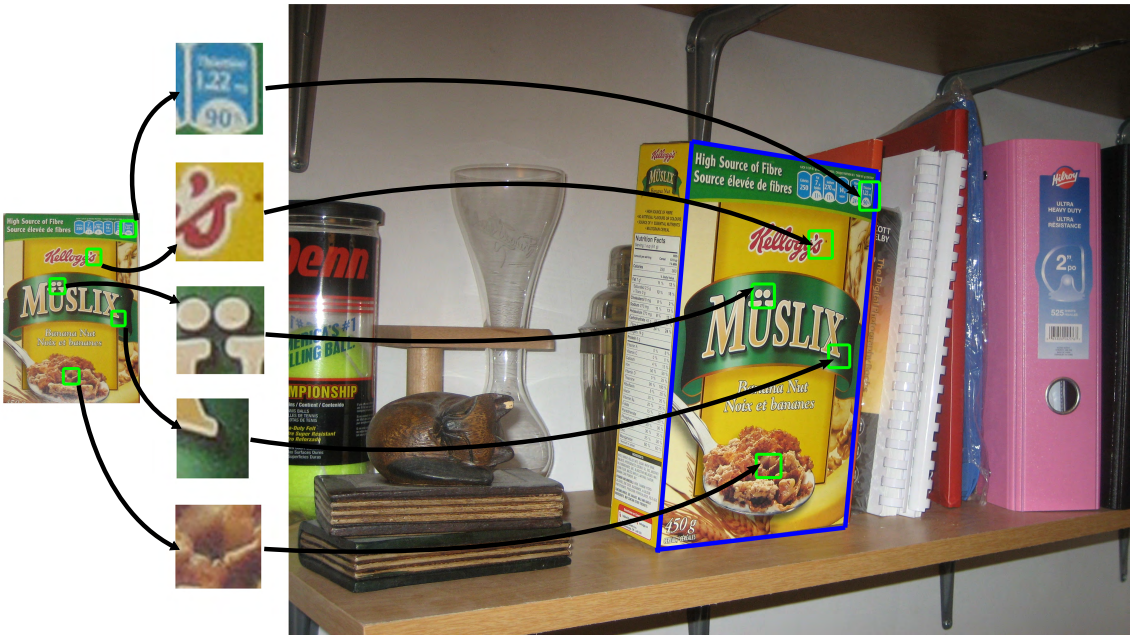


Figure 2.3 Feature-based object recognition.

Instead of using a reference image as a template, we extract *interesting* patches from a training image of an object for example and find matches of that same object in a different image (or video stream). As will be seen later on, such an approach will remove many of the previously mentioned limitations found in template matching.

These algorithms can be divided into two main stages : keypoint localization (or extraction) and feature description. At the keypoint localization stage, we mainly try to extract

salient and “interesting” points in the image (in order to later on describe the patches surrounding these points). Keypoints can be thought of as edges, corners, or even scale invariant points.

As previously discussed in template matching, one could choose patches to be matched from one image to another ; a training image or set of images can be used for that purpose. The problem with that approach is that some patches are more salient, or contain more distinctive information, and thus can be matched more easily than others. Figure 2.4 shows an example of good patches in green and bad patches in red.

Figure 2.4 Feature matching - Good features (green) have stable locations and are distinctive. Bad features (red) can be easily mistaken for one another.

The red patches are not stable and are very similar in nature hence causing them to be very easily mistaken for one another.

The matching procedure naturally assume a certain amount of resemblance in between the images to be compared. From a first observation one could notice that if we sample patches around corner points, these would tend to be more distinctive or uniquely identifiable.

In terms of a mathematical definition, weak features, that cannot be easily matched, have

a low variance like a plain wall for example. One may then argue that edges may form good candidates. Looking at Figure 2.5, we can see, that even though edges do have significant variance, they tend to have it in a single direction (e.g. in the x-direction). This causes what is known as the aperture problem Todorovic (1996). The net result is that the motion vector of the feature would be ambiguous, and if we slide along the edge we may find many similar patches.

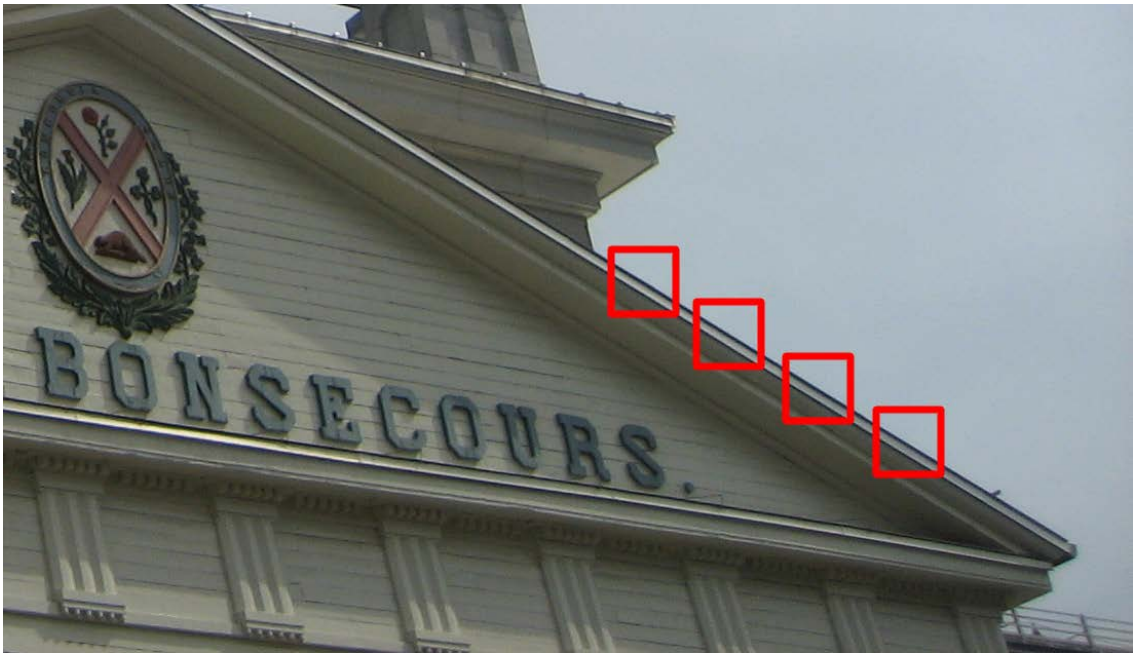


Figure 2.5 Edge features are bad since they tend to slide along the edge and are similar in nature.

Corners, on the other hand, would make for good features since, these have variations in all directions and hence will be relatively stable and can be extracted robustly.

2.2.1 Corner Detectors

Feature based methods depend on corner or interest point extraction. Corners may have different mathematical definitions and some may be more robust in terms of rotation, scaling, affine transformations, and even determining textured or blob regions.

Harris Corner Detector

The Harris corner detector Harris and Stephens (1988) is one the most commonly noted methods for extracting corners in literature. In a nutshell it attempts to find windows around a point in the image that when translated slightly produces a large difference. The window

has to be moved in both the x, and y directions. From a mathematical perspective, the boils down to calculating the autocorrelation matrix of the second derivatives of the image as shown in the equation below :

$$M(x,y) = \begin{bmatrix} \sum\limits_{-K\leq i,j\leq K} w_{i,j} I_x^2(u,v) & \sum\limits_{-K\leq i,j\leq K} w_{i,j} I_x(u,v) I_y(u,v) \\ \sum\limits_{-K\leq i,j\leq K} w_{i,j} I_x(u,v) I_y(u,v) & \sum\limits_{-K\leq i,j\leq K} w_{i,j} I_y^2(u,v) \end{bmatrix} \tag{2.5}$$

Note that $u = x + i$ and $v = y + j$. The window size is defined using the K parameter. I_x and I_y respectively indicate the image derivatives in the x and y directions. A weighting window, typically a Gaussian one or a uniform one, can be applied to the equation using the $w_{i,j}$ term. The second derivative terms do not respond to uniform gradient changes and hence when the eigenvalues of the above matrix are considered, we end up with a quantitative signature value that is invariant to rotations Bradski and Kaehler (2008).

A cornerness score R can be calculated using the determinant and trace value of the M matrix, to determine whether a certain point qualifies as corner point.

$$R = det(M) - \alpha * trace(M)^2 \tag{2.6}$$

α in this case is a weighting factor and the score can then be compared to a fixed threshold value.

Shi-Tomasi

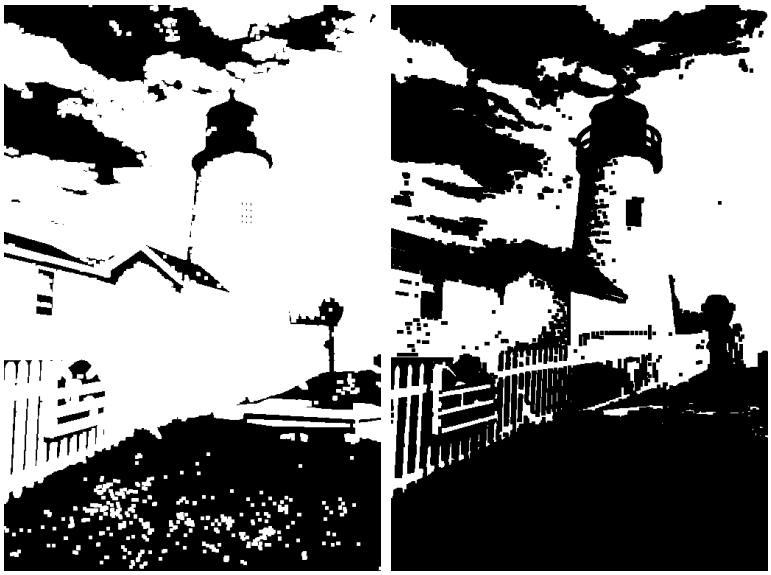
Shi and Tomasi presented a slight variation to the Harris corner detector which improved the performance and robustness of the method. Their method involved comparing the smaller eigenvalue to a fixed threshold, and it was proven to be enough to determine corners Shi and Tomasi (1994).

Morphological Corner Detection

Image morphology, whether in a grayscale or binarized context, can also provide an appealing approach to corner detection. These operators depend on the brightness level and are typically computationally simpler in nature as compared to the other methods while still being robust to noise Laganière (2011). Image morphology is typically formed of combinations of dilate (or maximum) and erode (or minimum) operations operating with pre-define mask structure or structuring element. Figure 2.6 shows erode and dilate operations being performed on binarized image.

Sequential combinations of these operators can be used to perform morphological close

(a) Original (b) Binary (Thresholded)



(c) Dilate (d) Erode

Figure 2.6 Morphological operators applied to a binarized image. 5×5 square structuring element were used in this case.

(dilate followed by erode) and open (erode followed by dilate). For the purpose of corner detection asymmetrical closing morphological elements Laganière (1998) are used like the ones shown in Figure 2.7.

The cornerness measure in this case would be calculated as such :

$$R = |I^c_{\text{cross,diamond}} - I^c_{\times,\text{box}}| \tag{2.7}$$

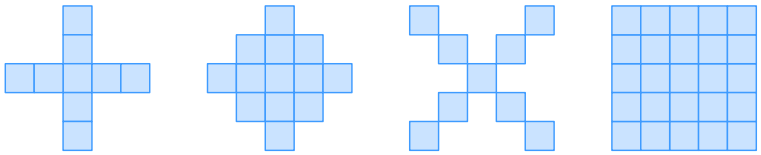


Figure 2.7 Morphological kernels used for corner detection : Cross ; Diamond ; X ; Box.

where I^c denotes a morphological close operation. Afterwards, the cornerness can be simply compared to a fixed threshold Laganière (2011). Note that sometimes this may lead to multiple pixel responses, not just a single corner location, hence non maxima suppression can be applied to avoid such conditions.

Multi-Scale Detectors

It is to be noted that the Harris detector is not scale invariant ; for that a multi-scale Harris detector is also found in the literature Lindeberg (1998). Instead of operating on a single image scale, the image is repetitively blurred to form a scale-space and the cornerness score is now compared with the finer and coarser scales. A corner would thus be an extremum.

Similar approaches, such as the Difference of Gaussians (DoG) that was used in the Scale Invariant Feature Transform (SIFT), are scale invariant and will be discussed in Section 4.2. In depth reviews and comparisons of such interest point detectors were performed in Mikolajczyk and Schmid (2005) Mikolajczyk and Schmid (2004).

2.2.2 Scale Invariant Feature Transform

Corner detectors, on their own, may be good enough to match features or keypoints across images with minor movements, for example for tracking purposes. But for the requirement of matching features or even whole objects in images with varying illumination, scale, and rotation, corner detectors alone fall short and hence feature-based method such as the Scale Invariant Feature Transform (SIFT) Lowe (2004) come into play ; These add a feature description or signature than enhances the matching and distinctiveness of features.

The SIFT algorithm is appealing to us since is it quite robust and able to match features with variations in scale, rotation, illumination, and even partial viewpoint changes that can be characterized as affine transformations. Moreover the features are distinctive enough that any given one can be matched against a large database, making it ideal for applications such as object detection, scene classification, and image registration.

An example of SIFT being used for an object recognition application is shown in Figure 2.8. The incoming video stream SIFT features are calculated by extracting “interesting”

keypoints and then describing the "patches" around these keypoints. Afterwards we try to match know objects from our database with the incoming keypoints. For the matching stage, a best-bin first (which is similar to a nearest neighbor search) is performed. The actual recognition stage is done by the use of RANSAC which would eliminate false matches which are considered as noise. In Figure 2.8 the purple lines and indicate correct matches whereas the red line shows an incorrect match. The final output would be able to estimate the pose of the object in the incoming video stream (shown as the blue rectangle).

The method can be divided into two major parts :

- Keypoint extraction : This part is comparable to a multi-scale corner extraction (or blob extraction).
- Keypoint description : Patches around a keypoint are chosen and characterized to be used as the patch signature.

Note that such algorithms tend to be computationally complex in nature due to the shear amount of data required to be processed. On the other hand, some parts of the algorithm are parallelizable and we will taking advantage of that as will be seen later on.

Keypoint Extraction

The keypoint extraction part can further be divided into smaller stages :

- Gaussian scale space (GSS)
- Difference of Gaussians (DoG)
- Keypoint localization
- Keypoint refinement

Gaussian Scale Space

Image pyramids as shown in Figure 2.9 are a common practice in the image processing world Adelson *et al.* (1984) (image pyramid some history) and are typically used to model scale variations. The image is progressively blurred to generate *scales* and then subsampled to form a new *octave*.

The 2D Gaussian filter has the following form :

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left[\frac{x^2+y^2}{2\sigma^2}\right]} \tag{2.8}$$

The standard deviation σ defines the blurring factor. Moreover in order to use the Gaussian kernel in a digital signal or image processing application, the kernel is sampled and the

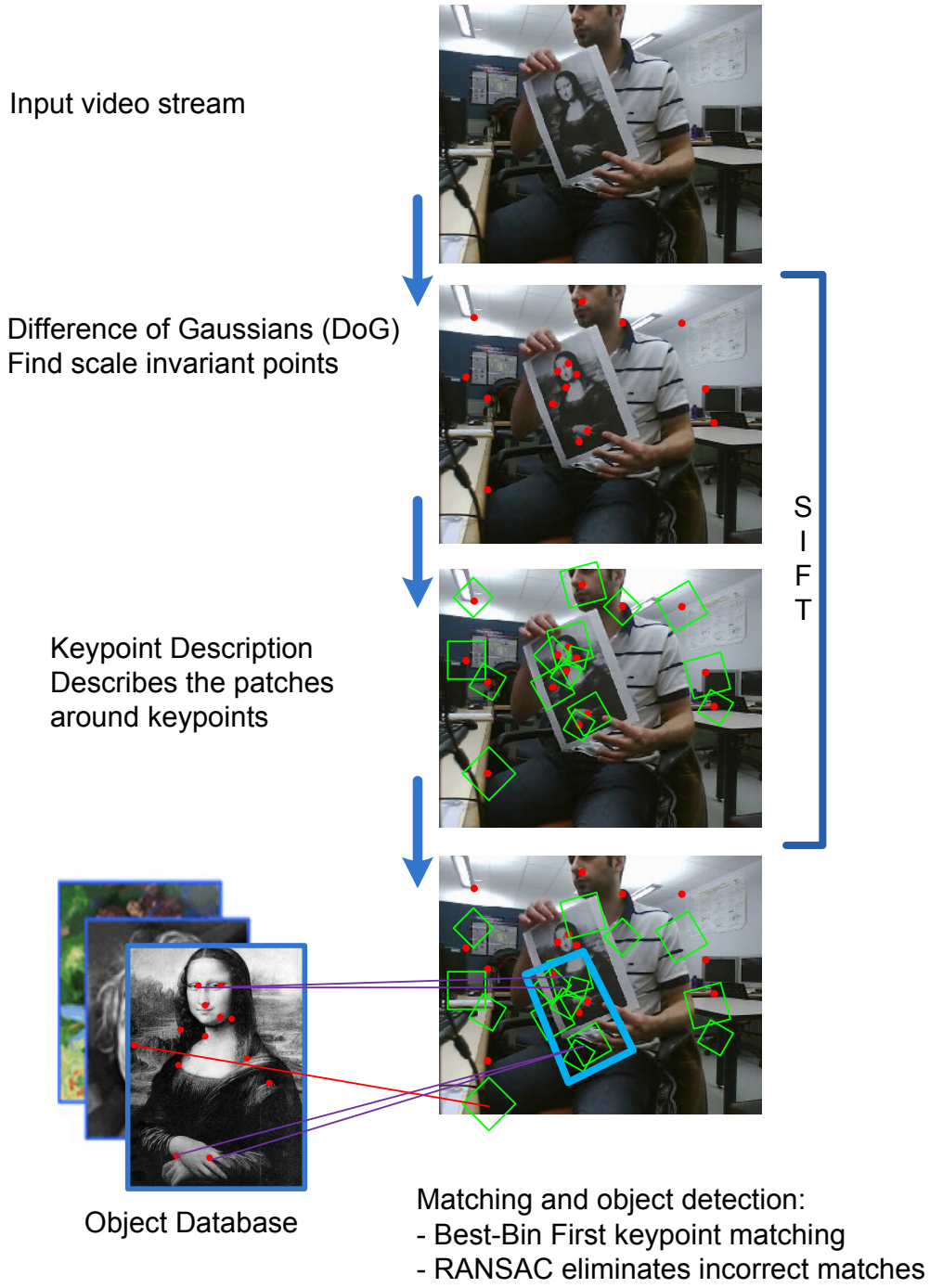


Figure 2.8 SIFT example : Object recognition

window is truncated and hence be applied as a Finite Impulse Response (FIR) filter. To preserve accuracy, the kernel window can be chosen as a function of σ.

Figure 2.10 shows a Gaussian Scale Space (GSS) image pyramid.

Such a pyramid can be either calculated by repetitively blurring the image or blurring

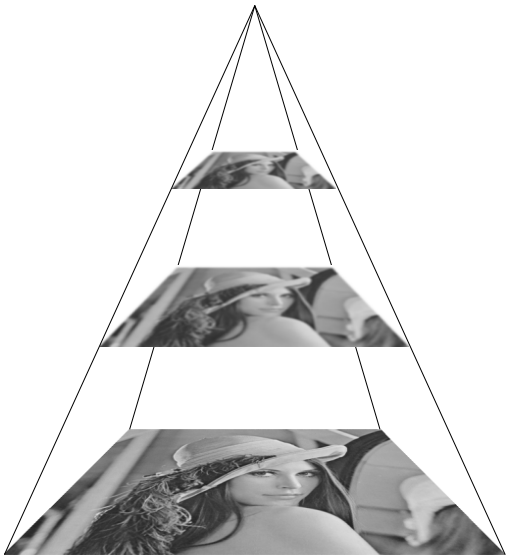


Figure 2.9 Image pyramid.

Figure 2.10 Gaussian Scale Space (GSS).

the original with an increasing σ value. Notice how the image loses detail as the blurring level increases through scales. The Gaussian blurring is at its base a low pass filter, thus after a certain amount of blurring, the high frequencies are eliminated and some pixels become redundant hence subsampling is performed without causing information loss.

Difference of Gaussians

At this point it would be pertinent to mention the Laplacian of Gaussians (LoG) Lindeberg (1994), Lindeberg (1999) which is one of the most common blob detectors. Essentially the LoG is second order derivative of the image ; such an operation is sensitive to noise so in

practice is it preceded by a Gaussian filter to reduce spurious effects. The operation responds to corner points and edges.

The LoG is computationally expensive and here is where the Difference of Gaussians (DoG) operation comes into play, Lowe (2004). The DoG is calculated by subtracting consecutive scale images in the GSS from each other and provides a good approximation to the LoG. Moreover the LoG at its base is not scale invariant and extending it to be so would involve increasing the complexity. The DoG on the other hand involve a simple subtraction of the pre-calculated scales and would provide us with scale invariant points which are better for tracking and matching.

Another point worth noting is that the scale images resulting from the GSS can be reused in the keypoint description stage of the algorithm.

Figure 2.11 shows some sample images resulting from the DoG operation. These images are scaled to the range $[0, 255]$ for visualization.

Figure 2.11 Difference of Gaussians (DoG).

Keypoint Localization

After having calculated the DoG images, we need to find the extrema points to determine the points of interest. This is done by comparing a pixel to its direct neighbors (same scale) and the neighbors in the finer and coarser scale. $3 \times 3 \times 3$ comparators are used for that purpose (compares a central pixel to its 26 neighbors), if a maximum or minimum is found, the (x, y, s) coordinates are considered as a point of interest. Note that the scale s is also kept into account at this stage since it will be used for the keypoint description stage that follows.

Since only scales from a given octave can be compared to one another (having the same

image dimensions), the first and last scale in a GSS octave are redundant from one octave to another in order to be able to transition smoothly through octaves.

Keypoint Refinement

The previous DoG and $3 \times 3 \times 3$ comparators serve as a blob detector. The SIFT algorithm was further refined as such : the corners that are found by preforming a sub-pixel localization ; this is done by fitting a 3D quadratic function to the local pixel region, Brown and Lowe (2002).

Some of the blobs can further be eliminated by checking if they have low contrast and if the blob is located on an edge instead of a corner. The corner checking part was borrowed from the Harris corner detector Harris and Stephens (1988). The difference is that the function will now only have to be applied to small regions of interest instead of the whole image.

Keypoint Description

After having extracted stable keypoints at known scales, we now have to build the keypoint descriptor. This stage can also be divided into finer parts as such :

- Orientation assignment
- Description vector

Orientation Assignment

The image gradients are used instead of the actual brightness values in order to be illumination independent. Image gradients are defined by a magnitude and angle as shown below :

$$\begin{aligned} m(x,y) &= \sqrt{(I(x+1,y) - I(x-1,y))^2 + (I(x,y+1) - I(x,y-1))^2} \\ \Theta(x,y) &= \tan^{-1} \frac{I(x,y+1) - I(x,y-1)}{I(x+1,y) - I(x-1,y)} \end{aligned} \tag{2.9}$$

Keypoints at this point are scale invariant, but they still have to be formulated to be rotationally invariant. For that, we choose a certain region around the keypoint usually define by a 16×16 window. The orientation gradients are collected and binned into 36 bins, the magnitude value sets the amount that has to be accumulated in a given bin. The highest value in the histogram is used to determine the dominant orientation of the keypoint.

The orientation is used for the given keypoint and the description vector that will be generated in the final stage is normalized according to this dominant orientation.

Description Vector

The final part of the algorithm consists of generating a distinctive signature or feature vector to a given keypoint. One has to keep in mind that the image gradients around a point, even if they are rotationally normalized, will never provide a perfect match. In order to accommodate for slight variances, the SIFT algorithm rather subdivides the 16×16 window into smaller 4×4 sub windows and calculates the mini-histograms of orientation gradients of each of these regions as shown in Figure 2.12.

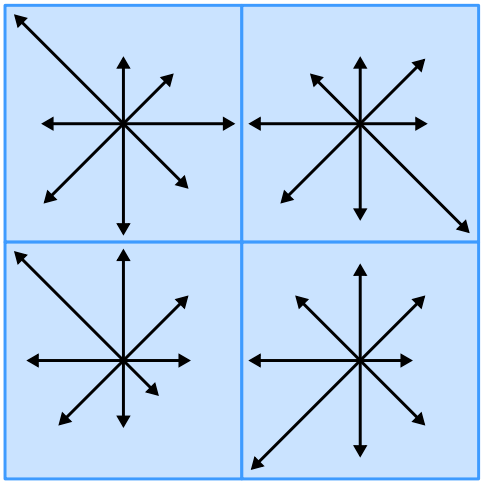


Figure 2.12 SIFT descriptor. Note : The figure shows a sample 2×2 grid of 8 bin histograms, whereas the actual algorithm uses a 4×4 grid.

These histograms use 8 orientation bins and give a certain error tolerance and variation robustness to the descriptor. Moreover the large window is weighted by a Gaussian smoothing factor, this way gradients closer to the keypoint center contribute more significantly. The final output of the description stage is a $4 \times 4 \times 8 = 128$ vector that describes the small 8 bin histograms around a keypoint. In order to be orientation invariant, the major orientation is subtracted from the vector and the vector is also normalized.

2.2.3 Other Feature-Based Algorithms

Other feature-based algorithms are available in the literature ; One of the most noted ones is the Speeded-Up Robust Features (SURF), Bay *et al.* (2008), Bay *et al.* (2006), which is inspired from the SIFT algorithm. SURF attempts to reduce the computational complexity by making use of integral images. Integral images were first used in Viola and Jones (2004) to provide a fast way of performing simple large averaging convolutions. SURF makes uses of box filters to achieve an approximation of the Hessian matrix to perform scale-based

corner detection, the authors label their detector as the Fast-Hessian detector. A box filtering operation will require only 3 addition/subtraction operations to calculate the average of a rectangular region independent of the filter size (assuming that the integral image is pre-computed). This is quite advantageous for large filters and removes the requirement of using image pyramids. The GSS pyramid, that was discussed in the SIFT part, could now be replaced by several filters of increasing size. Note that such calculations will still be sensitive to high frequency variation due to the box filter's nature. After determining the corner points the SURF method proceeds in a manner similar to SIFT by a calculating a dominant orientation to the keypoint and binning the gradients into either a 64 or 128 dimensional vector. SURF also has an upright version, U-SURF, which reduces calculations and only works for upright features. Furthermore, a comparative study Juan and Gwun (2009) shows that the SIFT descriptor remains more repeatable especially with large rotation differences.

PCA-SIFT, Ke and Sukthankar (2004) is another contender that is worth mentioning. Instead of using SIFT's weighted histogram to form the keypoint description, PCA-SIFT uses Principal Component Analysis (PCA) to normalize the gradient patch and reduce the dimensionality of the descriptor to 36 bins. The idea behind it is to make features more robust to deformations and due to the smaller feature vector size the matching stage could be done faster. The down side of such an approach is that although the matching might be accelerated, the feature computation becomes more complex.

Gradient Location and Orientation Histogram (GLOH), Mikolajczyk and Schmid (2005) is yet another SIFT variant, although the descriptor is more distinctive, is also makes use of PCA which again increases the calculation complexity.

A slightly different approach to SIFT is presented in Dalal and Triggs (2005). The main difference stems from the fact that instead of using a corner detector, the features are calculated on a dense grid of uniformly spaced cells. HOG's typical application is in pedestrian detection, the fallbacks of this method is its inability to support object rotation and hence assumes that a person will remain in a more or less upright position.

2.3 Image Processing Techniques in FPGAs

An FPGA is a highly configurable device that gives several degrees of freedom in order to achieve many given functions and is well suited for DSP algorithms, Goslin (1995). One of the first questions that we would encounter is whether to parallelize our design or serialize it at a higher clock rate in order to reach the required throughput ? The answer is best explained as a tradeoff between saving resources vs. meeting timing constraints.

As the parallelism increases, so does the resource consumption ; whereas serializing a

design (while maintaining the same throughput) would imply increasing the clock frequency hence making it harder to maintain the timing constraints.

Note : Mixed approaches can also be employed in which the parallelism is reduced (not necessarily fully serialized). Also different parts of a given algorithm can be implemented at different parallelism and/or clock frequencies depending on the data sampling rate.

2.3.1 Image Protocols and Interfaces

Efficient coding practices such as the ones presented in Garrault and Philofsky (2005) will accelerate designs and map efficiently on selected platforms. Typically, image processing in FPGAs is performed in raster scan order, i.e. the pixels are streamed in sequentially from the top-left corner to the bottom right. Additionally, due to the electronic nature of the processing, there might be some idle clock cycles were no valid pixels are transferred. In order to target the more general case, one would usually tend to define a series of signals or an image protocol to define the image content and its validity at any point in time. For that, signals such as the ones shown below can be used :

- Valid : Indicates whether the current data packet is valid.
- End of Line (EoL) : Indicates the end of the current line.
- End of Frame (EoF) : Indicates the end of the current frame.

In order to have reusable and standalone components, input and output FIFO interfaces can be employed. This would imply the need to implement proper flow inside the components to guarantee data integrity. Figure 2.13 shows an operator (or module) with a generic FIFO interface. Note that the black arrows represent data flow whereas the blue arrows represent flow control signals.

Figure 2.13 Module with a generic FIFO interface.

Concerning the actual pixel or data types that are used, the conventional approach would be to use a fixed point representation to perform arithmetic operation Meyer-Baese (2004) whereas floating point would be reserved for applications that critically depend upon it.

2.3.2 Types of FPGA Image Processing Operators

Due to the fact that the pixel data is being streamed in and the fact that FPGA devices are typically limited in terms of internal memory, one could divide the image processing operators into three major categories :

– Pixel-based.
– Neighborhood.
– Frame-based.

Pixel-based Operators

Pixel-based operators only depend on the information found in the current pixel and hence do not require any buffering or memory elements for the incoming pixels. Hence the general case of such operators would be of the form :

$$J(x, y) = f(I(x, y)) \tag{2.10}$$

where the current output pixel $J(x, y)$ is solely a function of the input pixel at that same coordinate $I(x, y)$. Examples of such operators are given below :

– Color space conversions (e.g. RGB to HSV and RGB to YUV)
– Color Correction : This involves linear or non-linear matrix multiplication of the color elements usually by matrices of 3x3 or 3x10.
– Flat Field Correction (FFC) : a gain and offset are applied to each pixel in order to correct the image sensor's non-uniformity.
– Arithmetic functions : Arctan, division, multiplication, addition, subtraction.
– Look-up tables (LUT).

The output and separate components of the YUV are shown in Figure 2.14.

Note that the YUV space is typically used in some compression algorithms. The main reason behind is that the human visual system is more sensitive to edge and grayscale variations. The Y component carries that information and hence the U and V components can typically be subsampled while still maintaining the main gist of the image. Moreover, the Y component can be used for grayscale processing (as the one our application would be based upon).

In many cases the YUV and YCrCb terms may be used interchangeably. The images in Figure 2.14 were calculated using Equation 2.11 ; Every output pixel is only dependent on the current input pixel data and hence the raster output can be preserved without any buffering requirements.

$$\begin{aligned} Y &= 0.299 * R + 0.587 * G + 0.114 * B \\ U &= 0.492 * (B - Y) \\ V &= 0.877 * (R - Y) \end{aligned} \tag{2.11}$$

Some functions, such as the FFC and LUT, may fall into a gray region, these require

(a) YUV (b) Y

(c) U (d) V

Figure 2.14 RGB to YUV output

buffering some pre-defined coefficients, but again the input pixel stream is not being buffered. Figure 2.15 shows a pixel-based FFC operation. I indicates the input image stream, A denotes the address (typically depends on the x and y coordinates of the current pixel, and O, G, and J respectively denote the offset, gain, and output images.

Note that the FFC look-up table, highlighted in blue in Figure 2.15, indicates the requirement of a memory element, typically in the case of 2D images the FFC coefficients will have to be stored in external RAM (due to the limited number of BRAMs in an FPGA).

Alternatively, for 1D images (line-scan cameras) the FFC coefficients only span a single line and storing them in BRAM maybe actually be a feasible and preferred alternative.

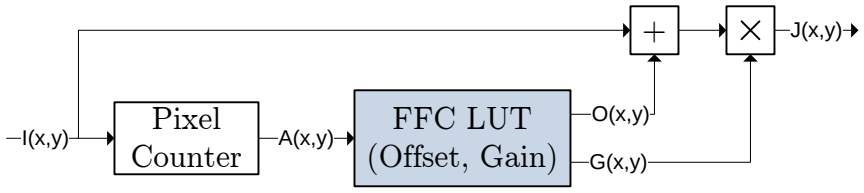


Figure 2.15 Flat Field Correction - Pixel-based.

In order to achieve the higher clock rates, many of these operators are highly pipelined like the Arctan function which is based on the CORDIC algorithm.

Neighborhood Operators

Neighborhood operators differ from the pixel-based ones in that a region or sliding window is required to calculate the current output pixel. It can formulated in a general form as shown in Equation 2.12.

$$J(x,y) \;=\; f(I(x+m, y+n)) \;\text{ where }\; \begin{array}{l} m = [-wx, -wx+1, ..., wx] \\ n = [-wy, -wy+1, ..., wy] \end{array} \tag{2.12}$$

Examples of such operators are given below :

- FIR filters (e.g. Gaussian, Sobel, ...)
- Median filtering
- Morphology (e.g. Erode, Dilate, Open, Close, ...)
- Bayer conversion (grayscale to color)

Keeping in mind that our pixels are being streamed in a raster scan manner ; in order to correctly buffer a window of pixels (or a kernel), we will need to store several lines of the image as shown in Figure 2.16. We will denote this moving window operator as a "kernelizer" ; it is basically a long shift register. The blue components in the figure are registers that are simultaneously output kernels, and hence a *kernel stream* would be formed from the incoming pixel stream ; This could be fed to a filter core that performs a pipelined parallel multiply-accumulate operation.

Assuming that the following variables W_K, H_K, W_I, H_I, and *bpp* respectively denote the kernel width, kernel height, image width, image height, and bits per pixel ; It should be noted that the memory requirements of such an operator is highly dependent on the W_I and the H_K. This is due to the fact that the required number of buffered lines is equal to $H_K - 1$

and the line buffer size directly depends on the image width W_I. The actual memory bits (disregarding the output registers) would be equal to :

$$\text{Total Memory Bits} = (H_K - 1) * W_I * bpp \tag{2.13}$$

Typically, the long line buffers are implemented as cyclic RAM buffers which implement a long line of fixed or even programmable latency. Note, in the case of a dynamically programmable latency, one should define a maximum image width that has to be respected.

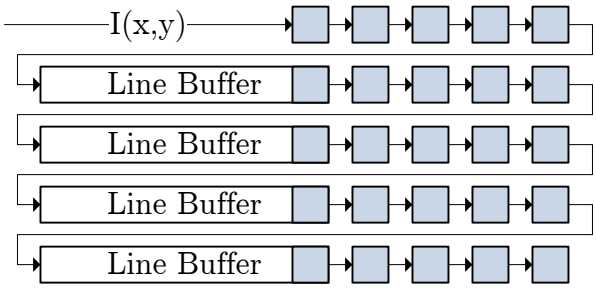


Figure 2.16 [5×5] Kernelizer. Line buffers are used to form a moving window around the required pixel

In the case of small (or relatively narrow images, one may opt to implement the line buffers using shift registers. The current Xilinx devices contain the SRL16E blocks that can be configured to use a single LUT as a 16-bit shift register Chapman (2000). Several SRL16E blocks can also be cascaded into a programmable addressable shift register.

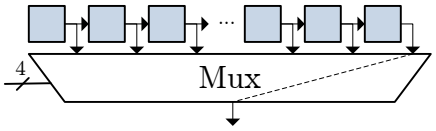


Figure 2.17 SRL16E - LUT can be configured as a shift register (maximum 16 latency).

It should be noted that the image height has no effect of the resource utilization, while the kernel width and/or parallelism will only slightly affect the FPGA register utilization.

The details of such a kernelizer operator can be implemented in different fashions. Keeping in mind that the operator has to flush the buffered pixels at the end of frame, i.e. either the input has to stalled (for that a frame buffer has to precede the operator) or the kernelizer would have to buffer an extra valid bit in order to perform a flush that does not stall the input ; Borders will also have to be handled as mentioned in Section 2.1.1.

Frame-based

Frame-based operators typically require buffering an incoming frame (or even several frames) to an external RAM as shown in Figure 2.18. In general, current FPGA devices are

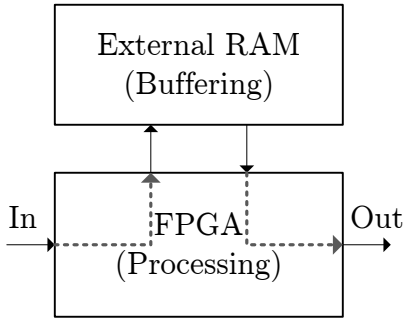


Figure 2.18 FPGA external RAM buffering.

only able to hold several lines of an image at a given time but do not have the required memory capacity to store whole frames (except if the images are relatively small and/or have a low number of *bpp*).

Examples of such operators are given below :

- Frame buffer (to decouple the camera source from the processing).
- Affine transform
- Multi-frame processing (e.g. Motion detection)
- Multi-camera synchronization (e.g. stereo vision) (see Figure 2.19)

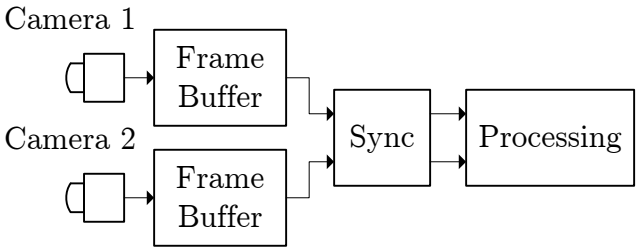


Figure 2.19 Multi-camera synchronization

A rotation operator (which falls under the category of an affine transform) is described in Figure 2.20 ; a forward mapping rotation approach is presented in Chen *et al.* (1999), it provides insight into a method to reduce memory accesses, but typically a more conservative backward mapping technique is used. Note that the output image is in raster scan order and hence would have to access pixels, from the input frame, in a manner that cannot be confined

to a window but rather requires the whole frame to be buffered. Also multiple pixels might be required and read in order to calculate a single output pixel, which is the case for when interpolating (e.g. bi-linear, bi-cubic) pixels that do not fall on an integer (x, y) coordinate of the original image.

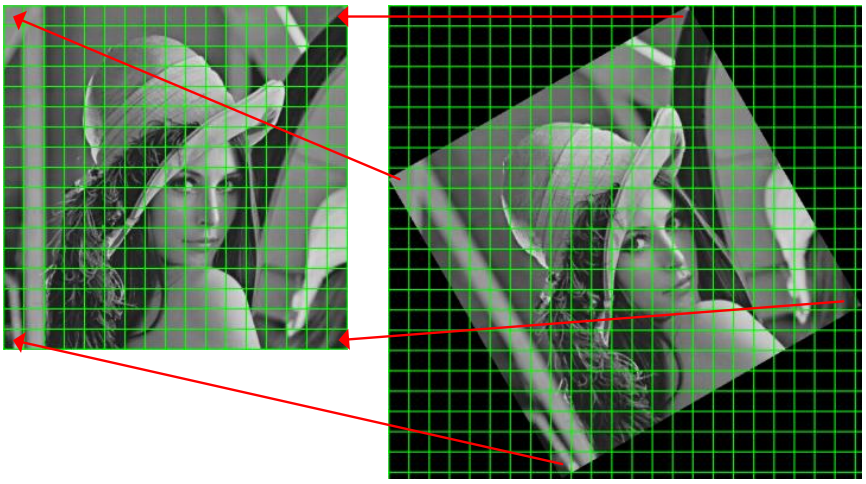


Figure 2.20 Image rotation - Backward mapping

It may be argued that for a certain fixed rotation angle, only a limited number of lines may be required, but for a 90° rotation the whole input frame will have to be buffered.

Frame-based operators is a rather generalized grouping since operators may differ in the way they actually read or address the external memory. In other words some functions would only read addresses sequentially (e.g. frame buffer) while others will perform "random" reads/writes to the memory which can seriously impact performance depending on the memory type being used.

Some commonly used forms of external memory components are DDR SDRAM, SRAM, and RLD RAM. DDR SDRAM will tend to have a high capacity, high bandwidth, and lower cost per byte, but such devices have long latencies and their performance will highly degrade with random addressing (which would cause the component to constantly change its active page) ; these devices are suited for frame buffers where line bursts (sequential addresses) can be written and read.

On the other end of the spectrum is the SRAM, which does not need to be periodically refreshed. This kind of RAM has a high bandwidth as well a low latency and it is well suited to perform random accesses ; such components are expensive and would have a significantly lower capacity than the DDR SDRAM.

As a third alternative, RLDRAM can be employed ; it provides a middle ground between

DDR SDRAM and the SRAM. RLDRAM capacities are a bit lower than the DDR SDRAM counterparts while having lower latencies and hence being more suited for pseudo-random accesses.

Depending on the required application (e.g. frame buffering, rotation, multiple small ROIs), required throughput, and cost, one would have to choose the most appropriate type of external RAM.

2.3.3 Finite Impulse Response

Image filtering operations, such as Gaussian blurring, fall into the category of performing a convolution. Such types of convolutions can be performed in either the frequency, or time domain. For most image processing FPGA applications using the 2D Fast Fourier Transform (FFT) will typically be expensive due to the shear computing and frame buffering resources that will be required and hence using a Finite Impulse Response filter would typically be used. FIR filters are linear time invariant, the kernels size of such filters can be adequately truncated depending on the filtering function.

FIR filters perform a convolution as shown below :

$$y[n] = \sum_{k=0}^{N-1} h_k x[n-k] \tag{2.14}$$

where x, y and h respectively represent the input signal, output signal, and filter coefficients. The signal in our case is the processed image.

These types of filters require the use of constant coefficient multiplications. Different hardware architectures can be used to implement a FIR, Hawkes (2005) as stated below :

- Direct Form
- Transposed Form
- Systolic
- Multiply Accumulate (MACC)
- Semi-parallel

A typical implementation of a FIR is shown in Figure 2.21, this makes use of the direct form. We may notice the use of an adder tree at this point to perform the summation. Although such an approach is simple to understand it does not always scale and/or fit nicely into newer FPGA devices that contain dedicated multiplier blocks (or multiply accumulators) such as the Xilinx DSP48E. Note that these dedicated multipliers are able to run at very high speeds (around $500MHz$, Patel (2005) depending on the device) as compared to the rest of the FPGA fabric.

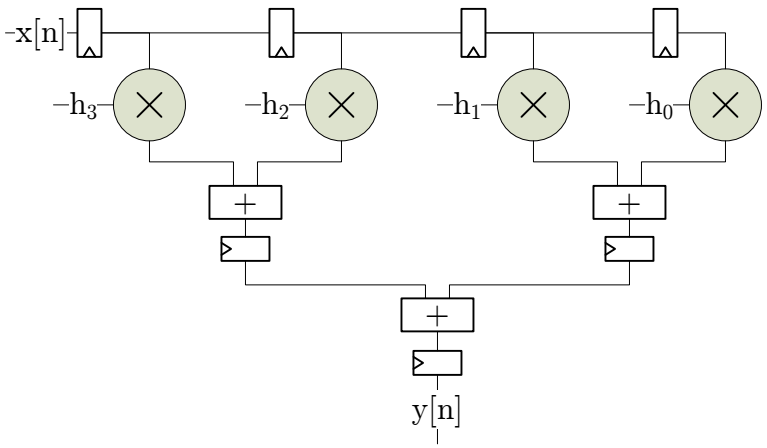


Figure 2.21 FIR - Direct form using an adder tree

A transpose form FIR is shown in Figure 2.22. As can be seen, this architecture fits nicely into the multiply accumulate blocks which are individually indicated by the dotted boxes. This type of filter has a rather low latency, but suffers from a high fan-out input which may be problematic for larger filters.

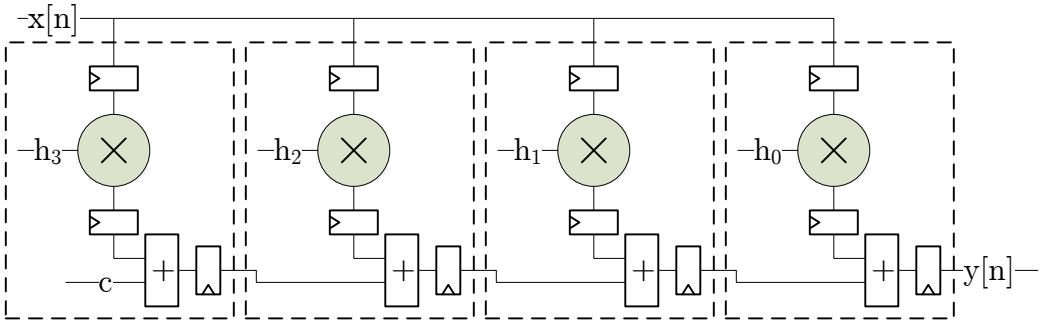


Figure 2.22 FIR - Transpose form

The systolic form shown in Figure 2.23, is adequate for high throughput and does not suffer from the large fan-out problem, but on the other hand it has a high latency which increases with the filter size. Note that in the actual implementation, the number of dedicated multipliers found in an FPGA column may actually affect the final timing performance that can be achieved due to the dedicated cascade connections.

The previously mentioned filters lend themselves to a fully parallel design where $SampleRate \leq ClockFrequency$. One would typically calculate his or her requirements based on the data sample rate, the actual clock frequency, and the number of taps used. One the other side of the spectrum, we may find the Multiply Accumulate (MACC) filters similar to the one shown in

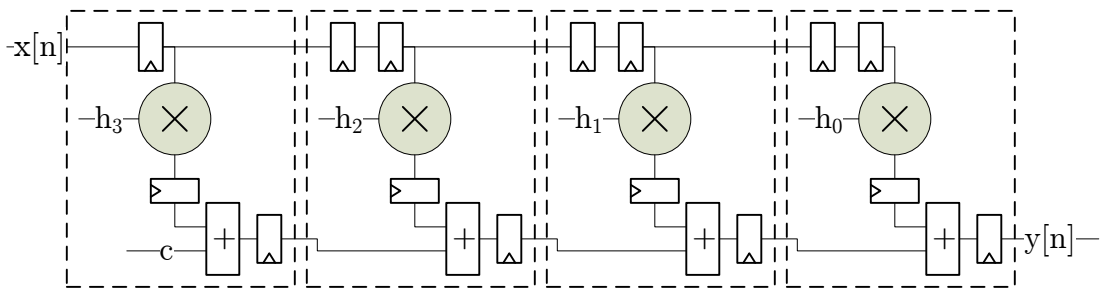


Figure 2.23 FIR - Systolic form

Figure 2.24. Such filters depend on the fact the $SampleRate \times NumTaps \leq ClockFrequency$. Notice how this filter saves resources by using a single multiply accumulate block ; The coefficients and incoming data samples are stored in a small RAM that can be appropriately addressed using a small controller/counter. Note that the accumulator in the DSP48E blocks

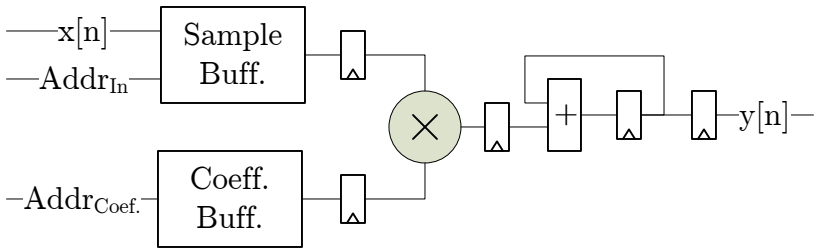


Figure 2.24 FIR - Multiply Accumulate (MACC)

allows a maximum of 48 bits to be stored as a fixed point signed number format, and if the results exceed this limitation, multiple block have to be cascaded together to perform that same function.

Semi-parallel architectures can also be formed using the above mentioned concepts ; These would provide a speed/resource usage tradeoff depending on the requirements.

In many cases the required filtering operations are symmetric, such as a Gaussian filtering, and for that one can make use of this to half the number of required multipliers by using a pre-adder and performing a single multiplication for the redundant coefficients. Note that such an architecture is not applicable to the transpose form, hence the filter would typically be implemented as a symmetric systolic FIR.

Another important feature, that can be exploited for image processing applications, is that in many cases 2D filtering uses a separable kernel. 2D filter separability means that instead of using a FIR with $K \times K$ taps, we can perform the filtering in the horizontal direction

($K \times 1$ taps) followed by another filter in the vertical direction ($1 \times K$ taps), thus reducing the number of taps to $2K$. Gaussian filtering falls into that category of kernels.

Note in some cases multipliers can be saved by using coefficients that are powers of two i.e. 2^N. Such multiplications can be performed in hardware as a simple shift-left operation that does not cost any resources. In some cases coefficients can be approximated to fit this requirement.

CHAPTER 3

APPROACH AND ORGANIZATION

In this Chapter we will discuss the approach and organization of the research. We will be explaining how the article in Chapter 4 fits in the research paradigm and will map the undertaken steps to the objectives stated in the Introduction.

3.1 Software Prototype

After performing a comprehensive literature review of the object recognition algorithms we were able to determine that the SIFT algorithm is best suited for our applications in terms of robustness.

3.1.1 Image Processing Libraries

In order to implement image processing algorithms, several optimized libraries where explored, such as the MATLAB image processing toolbox MATLAB (2010), the JAVA based ImageJ, Abràmoff *et al.* (2004), and OpenCV (C/C++), Bradski (2000).

OpenCV was used for the SW implementation and validation due to the fact that it contains a large range of functions and algorithms, is highly optimized, and can be easily connected to almost any camera device.

3.1.2 SIFT

Several software implementations of the SIFT algorithm, Lowe (2004) can be found in the literature. The author distributes the binaries on his website for academic purposes. Other open source implementations are available such as the MATLAB/C implementation by Andrea Vedaldi which is now superseded by VLFEAT, Vedaldi and Fulkerson (2010). A rather flexible C/OpenCV implementation is provided in Hess (2010). We chose to use the latter one for the software prototype.

3.1.3 Keypoint Matching

In order to perform single object recognition, the SIFT keypoint extraction has to be performed on a training image as well as every incoming frame.

Due to the rather distinctive nature of the SIFT keypoint signature, every keypoint can be individually compared and matched using a nearest neighbor measure. The training image keypoints would then form a database for which we would try to find a matches.

A nearest neighbor match is basically a Euclidian distance measure. Instead of choosing a single global threshold to determine a match, a better match criterion is to use the ratio of the closest neighbor to the second-closest neighbor, Lowe (2004). If the ratio is greater than 0.8 the match is rejected.

NOTE : this comparison metric is necessary due to the fact that some keypoints are more distinctive than others.
Typically an exhaustive search can be performed in order to find correct matches ; But such an approach would prove to be very costly. Moreover KD-tree indexing can be performed to speed-up the search process, but again this does not scale well to spaces with more than 10 dimensionality.

Since the SIFT keypoints have a 128 dimensionality, it was shown in Beis and Lowe (1997) that a Best-Bin-First (BBF) algorithm can be used. The BBF is basically a modified KD-tree method to approximate a nearest neighbor search. the two main differences from a KD-tree is that the BBF performs backtracking using a priority based queue and a fixed number of candidates are searched before the search gives up (in our case 200 nearest neighbor candidates are explored).

BBF provides a speedup of approximately 2 orders of magnitude, Lowe (2004).

3.1.4 Object Localization

After performing individual keypoint matching, we now want to approximate the location and even pose of the object in question. The main issues at this point are the following :

- The object might undergo affine transformations (zooming, rotation, skew, etc...)
- Objects might be occluded
- Some keypoints are erroneously matched (false positives)
- Some keypoints are missing (false negatives)

In order to overcome the aforementioned issues, we need to approximate a perspective transformation matrix which would provide a good enough fit. For that, the Random Sample Consensus (RANSAC) algorithm, Fischler and Bolles (1981) was used.

Keeping in mind that every keypoint carries x-y coordinates, scale, and orientation information, these can be combined to check for a correct match. RANSAC is an iterative algorithm that will use a random subset of inliers to minimize an error function. Compared to least-squares method of fitting, RANSAC is robust to outliers and noisy data.

Note that the generalized Hough transform, Duda and Hart (1972) can also be used to approximate or "vote" for the best solution as was presented in Lowe (2004). RANSAC and Hough matching methods were compared in Se *et al.* (2002) which draws the conclusion that for distinctive features, RANSAC would be more computationally efficient.

3.1.5 Lucas-Kanade

The SIFT algorithm does not work in real-time on a host machine and will typically be able to find a valid match at every second. In order to have a smoothly running software prototype, we used the Lucas-Kanade (LK) method for optical flow, Lucas and Kanade (1981).

The LK method is an iterative one that attempts to find feature or corner matches from one frame to the next. The LK algorithm relies on three basic assumptions, Bradski and Kaehler (2008) :

- Constant brightness : pixel values of an object do not change much from one frame to the next.
- Small movements : Inter-frame movements are relatively small.
- Spatial coherence : Pixels that belong the same object tend to move together.

Inherently, the LK method relies on the features to be chosen at appropriate "seeding" locations such as corners or DoG points.

In order to integrate the LK tracker with the SIFT, we used the Boost C++ libraries for multithreading. The SIFT was running on a thread and during the computation for the next frame the LK tracker would be used to estimate the object movements from one frame to the next. The LK corner point would only be initialized inside the bounding box of the SIFT detected object and the RANSAC computation would be performed on the tracked features.

Figure 3.1 shows a snapshot of the software implementation being run on an Intel Pentium D 2.8GHz with 2GBytes of RAM. The camera used in this case is a PlayStation Eye which is capable of capturing up-to 640×480 pixels at a rate of 60 frames per second (FPS). The window on the right shows the SIFT training object being matched in the incoming stream. The purple lines indicate a match.

Note that incorrect matches do occur and are ignored by the RANSAC as false positives.

The window on the right of Figure 3.1 shows the Lucas-Kanade tracker. The tracker initializes corner points, shown in green, inside the bounding box of the object that has been found. The points are tracked from one frame to the next until the slower multithreaded SIFT algorithm comes back with a new object bounding box.

A drawback of using this method is that the bounding box determination would have a latency of several frames.

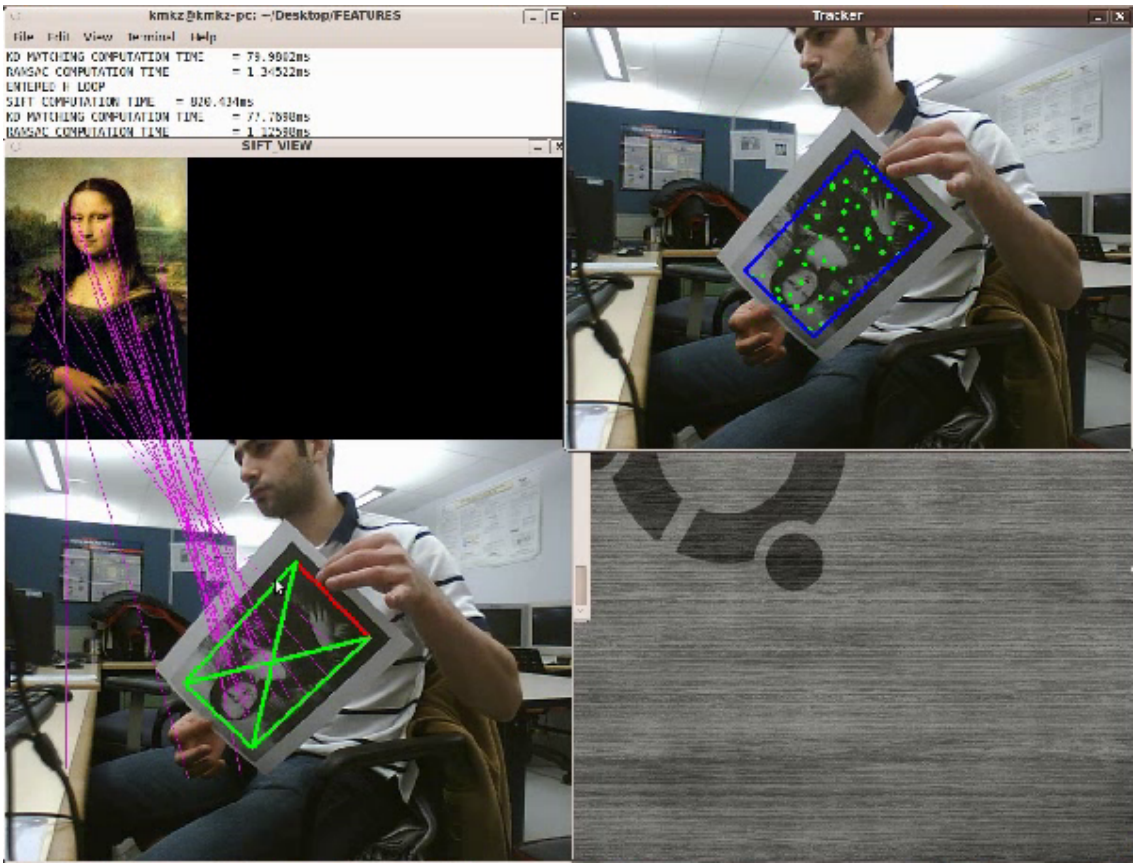


Figure 3.1 Snapshot of the multithreaded SIFT and Lucas-Kanade tracker.

3.2 Hardware Implementation

After performing a software evaluation of the overall system, we were able to prove that the system is robust enough to perform accurate localization and recognition of objects varying in pose and illumination. It also became apparent that the DoG part of the SIFT algorithm was the most computationally expensive.

In order to overcome that, a hardware approach was employed mainly due to the fact that the DoG is highly parallelizable. We presented a new pipelined and interleaved hardware architecture that can be used to compute the DoG image pyramid as will be seen in Chapter 4. The proposed architecture is efficient on resources and bandwidth, and was proven to be comparably accurate to a software floating point implementation.

3.2.1 Xilinx System Generator

The Xilinx System Generator for DSP was growing in popularity at the time, so it was well worth evaluating the possibility of using the tool for the hardware implementation.

System Generator integrates in MATLAB Simulink as additional hardware (Xilinx) blocks. Basically the user can switch almost seamlessly from the software to the hardware domain by using *Gateway In* and *Gateway Out* blocks that perform floating point to fixed point conversions and hence define the FPGA interface.
Figure 3.2 shows an example of how a basic inverter could be implemented in the FPGA fabric. Notice the *Gateway In/Out* ports that indicate the FPGA boundary. All of the available Simulink/MATLAB blocks can be used outside of the FPGA domain like the sources and sinks (scopes in this case).

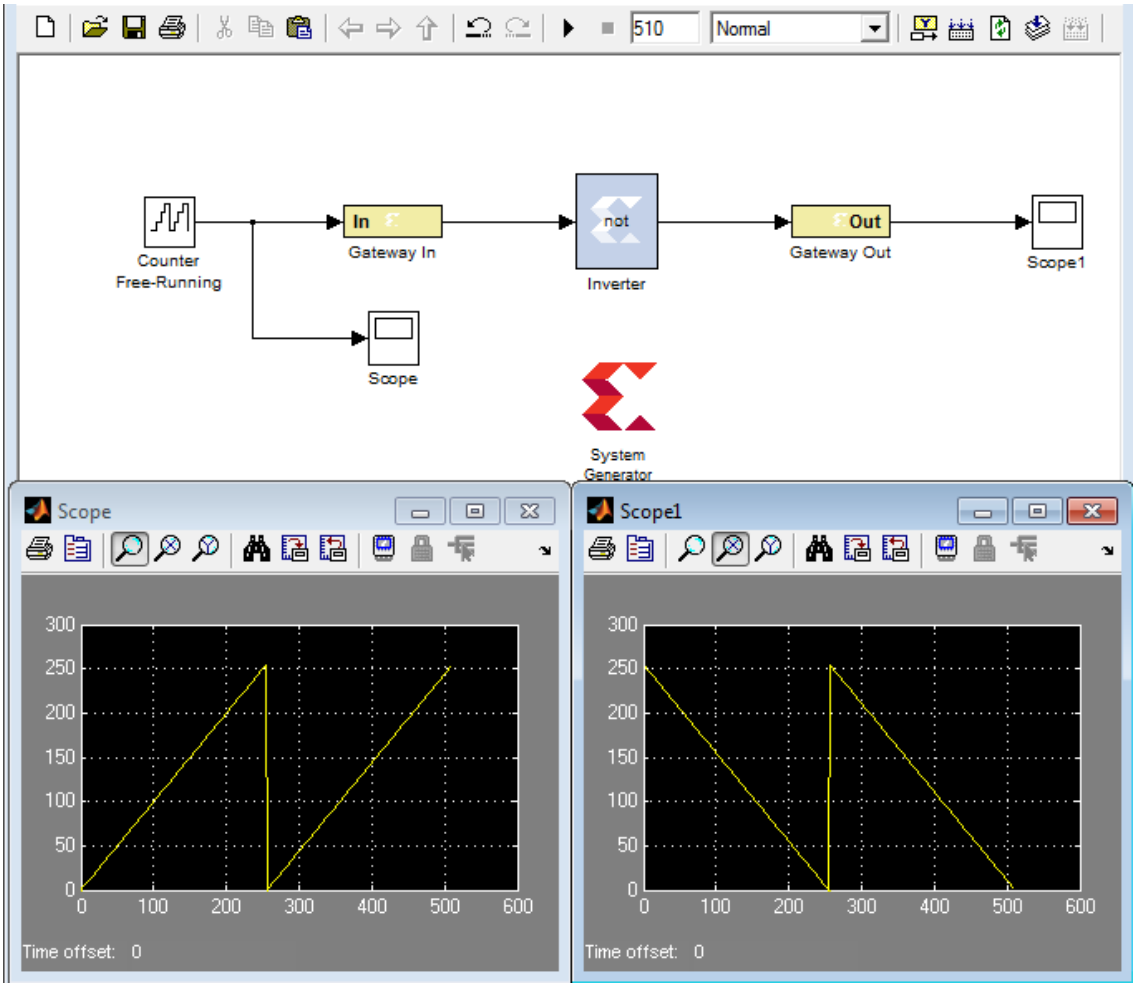


Figure 3.2 Xilinx System Generator - Basic inverter.

The Xilinx block set consists of both low-level and high-level blocks. The low level blocks would be mainly concerned with bit manipulations, counting, registering and synchronization, while some higher level blocks would be more concerned with more complex functions such as hardware Divider, FIR Compiler, Fast Fourier Transform (FFT).

The user may build his hardware functions into complex systems using block diagrams. Simulations can either be performed at the bit-level by connecting to an external hardware simulator such as Modelsim or Xilinx Isim, or even by having a software equivalent MATLAB implementation of the module which would tend to accelerate the simulation time.

One of the main features that makes System Generator attractive to the hardware designer is the ability to perform hardware co-simulation (hw-cosim), also known as simulation with hardware in the loop.

In our case, since we are using the Xilinx ML605 board, we were able to the Ethernet-based hw-cosim. Basically, the user can perform data block transfers using a simplified memory interface and/or a FIFO interface. The Xilinx tools would then generate the appropriate Ethernet controller to send the data from and to the host computer. Moreover, a register interface would also be automatically generate and controller from the Ethernet port to control any dynamic constants (registers) that may affect the data flow in the FPGA module.

Figure 3.3 shows a test bench of a VHDL function. Note that in this case a wrapper is used. Also the pixel data is propagated by the use of block transfers whereas other dynamic parameterizations and data alignment signals are communicated through the slower register interface.

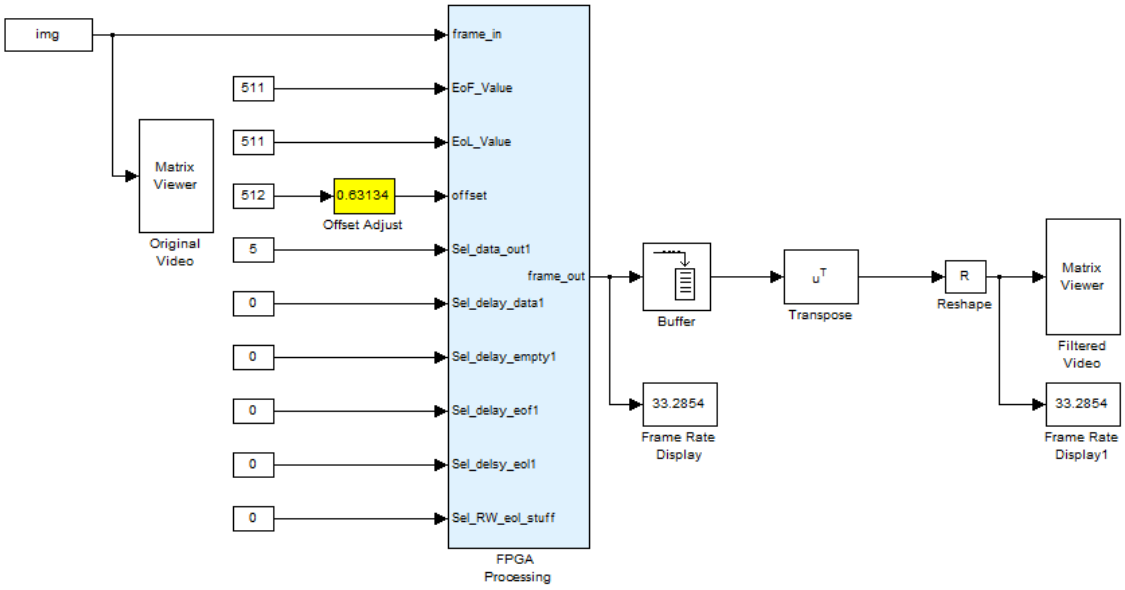


Figure 3.3 Hardware co-simulation test bench.

A wrapper is required to interface the VHDL module in the MATLAB/SysGen environment. Basically, all generics are hardcoded and the port names should be correctly mapped. Moreover, we add some simple logic to add some image control signals like the end of

line/frame and data valid. In addition to that we have added a debug multiplexer in order to be able to individually probe the image at each scale output.

Although using a schematic interface to insert and connect Xilinx components may simplify design entry and visualization, it introduces a limitation in that we cannot cascade and connect N blocks as we are able to do using a VHDL *for* generation loop.

In our case, we needed to construct a generic DoG hardware module that is highly parameterizable e.g. number of scales and octaves, number of filter taps, etc...

For that we favored using VHDL coding for the DoG implementation. The VHDL code can then be wrapped into a System Generator custom block that was imported and simulated as a hw-cosim module. System Generator was used as a test bench to validate the hardware modules.

The FIFO interface was used to perform the image data transfers. In order to optimize the transfer speeds, jumbo frames have to be used on the Ethernet adapter in question.

3.2.2 VHDL Coding

Having decided to code the DoG part in VHDL, in order to keep in line with our objectives, we kept in mind that the design has to be :

- Reusable : We should be able to easily switch platforms.
- Parameterizable : Depending on the final application different parameters may be used.
- Resource efficient : This would have the advantage of being able to have other functionality, smaller footprint, and even lower power consumption.

For that we started by coding all of our lower level components by inference ; such components would include multipliers, FIR filters, line buffers, memories, and synchronous FIFOs. More details concerning the FIR topologies are found in Chapter 2.
The final DoG implementation has the following parameters : maximum image width, number of scales, number of octaves, and number of filter taps. Moreover the architecture can be also modified to treat higher parallelism which would be able to process several pixels in a single clock cycle.

The number of scales and octaves are application dependent ; for applications that expect the objects to be relatively located at the same scales or zoom level, less resources can be utilized for that purpose. Parameters are cascaded through the use of VHDL generics and/or package declarations.

3.2.3 Architecture Optimization

In order to achieve a resource efficient design, we had to optimize the design from several aspects. The DoG inherently requires the repetitive Gaussian blurring and periodic subsampling of the image in order to produce the scales S and octaves O. A direct approach would imply using $O \times S$, 2D filter banks.

Data Interleaving

One thing to be noticed is that the filter bank coefficients repeat at every octave (scales have different coefficients). Also the data is reduced by a factor of 4 at every new octave due to the 2×2 subsampling. In order to take advantage of that, our architecture instead uses an interleaved approach hence reducing the number of filer banks to S. Octaves are interleaved hence producing an output at every other clock cycle as is shown in Chapter 4.

Gaussian Filtering

Due to the fact that a Gaussian kernel is used, we can take advantage of two distinctive properties. The first is the separability of the 2D Gaussian kernel; typically a 2D parallel hardware filter of M rows and N columns would require $M \times N$ taps. But in the case of a separable filter, an equivalent result can be produced by performing the filtering in the x-direction (horizontal) followed by a filter in the y-direction (vertical), hence reducing the total number of required taps to $M + N$.

The second property of a Gaussian filter is its symmetry. For that we can even half the size of our horizontal and vertical filters by using a pre-adder prior to performing the multiplication. This saves multiplier resources that are typically limited especially on the lower end devices or even in the case where multipliers are implemented using the FPGA fabric.

We have implemented scalable VHDL versions of the FIR that use multiply-add chains in order to be able parameterize the design using generics as will be seen in Chapter 4.

Kernel Buffer Sharing

A main issue of any cascaded image filtering design is the required amount of memory resources. BRAM is used to perform line buffers that can spatially align the incoming pixel stream prior to performing any kind of FIR operation. The image is typically fed in a raster scan order (top-left to bottom-right), hence a pure horizontal filter does not require any line buffers. On the other hand, any vertical or 2D filter would require buffering $M-1$ lines where M is the number of rows in the 2D kernel.

At every subsequent cascade stage, the data has to be buffered again in order to calculate a new scale output.

By noticing that, we came up with what we call the *sub-kernel trick*. Essentially instead of using a cascade of filters for the every scale, we use parallel filters that share the buffering resources.

Figure 3.4 shows how a $[5 \times 5]$ and $[3 \times 3]$ sub-kernel can be extracted from a $[7 \times 7]$ kernel (or moving window). The extracted parts are shaded in gray.

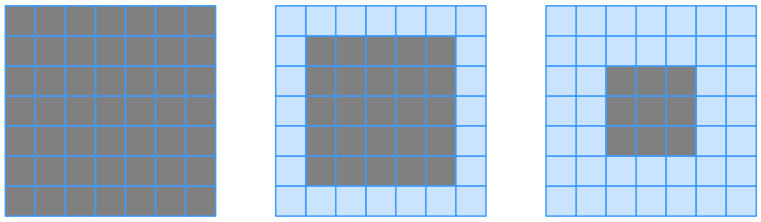


Figure 3.4 $[7 \times 7]$ Kernel. $[5 \times 5]$ and $[3 \times 3]$ Sub-kernels.

Note that the actual filter coefficients differ in the two approaches. The sub-kernel approach uses the direct Gaussian coefficients.

Figure 3.5 shows how a typical cascade of Gaussian filters can be implemented to calculate the different scales. Notice how the image has to be buffered at every new scale.

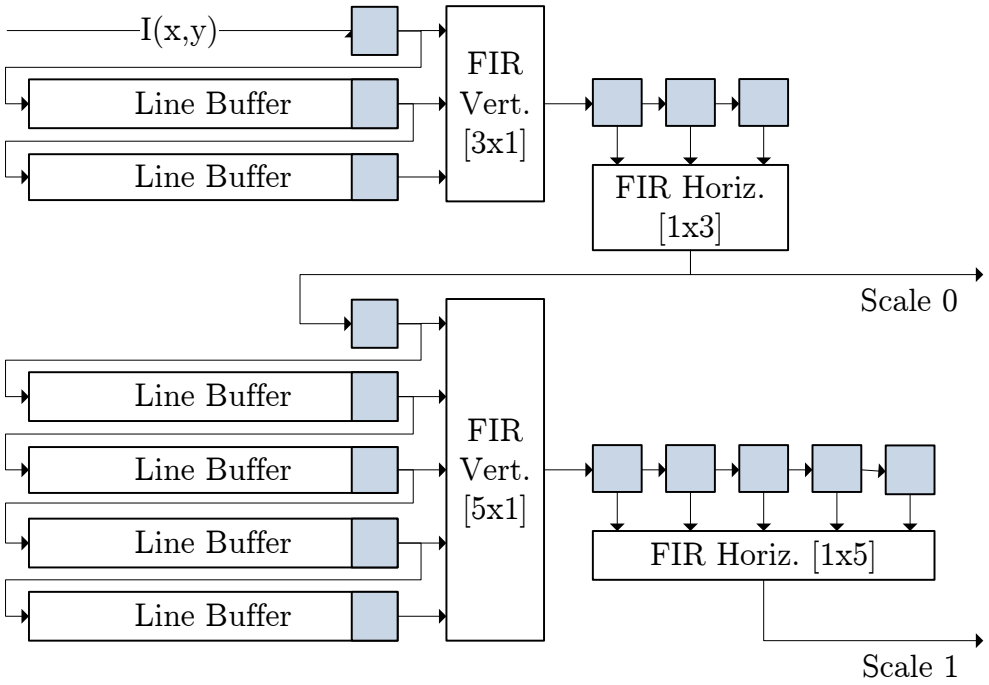


Figure 3.5 Cascaded approach for 2 scales.

The Gaussian FIR filters at higher scales have a larger standard deviation, and hence would typically require a larger number of taps to avoid abruptly truncating the filter kernel.

A key point is that the vertical filtering has to be performed prior to the horizontal one. By doing that we have the following advantages :

– BRAM resources are shared.

– Output streams of the different scales are aligned.

No extra synchronization pipelines or FIFOs are required prior to the DoG part where the outputs of the different scales have to be subtracted.

Figure 3.6 shows how the sub-kernel architecture differs from the cascaded architecture.

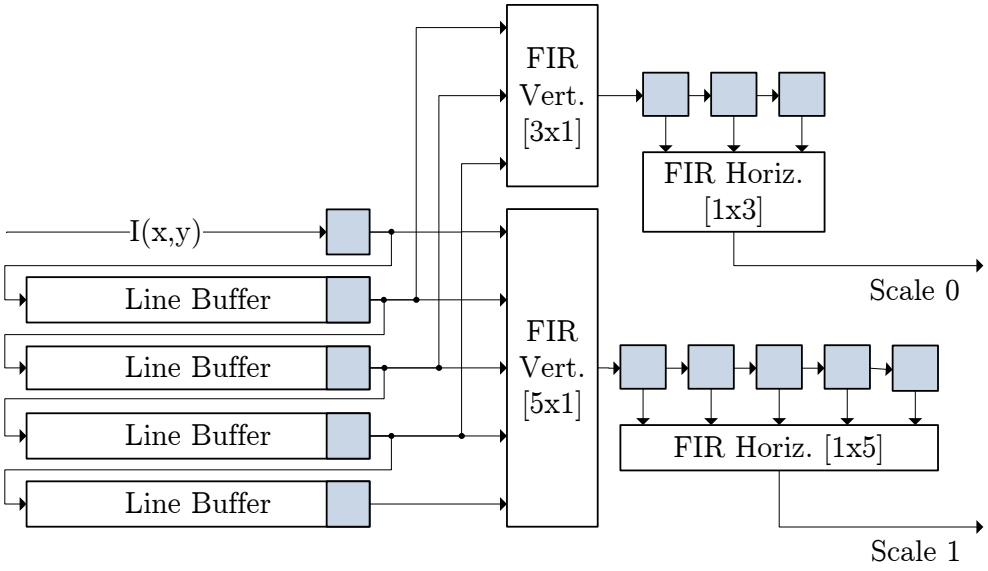


Figure 3.6 Sub-kernel approach for 2 scales.

Note that the horizontal filtering does not require BRAM buffering and would typically be implemented using the fabric registers.

Note that the octave data interleaving was also integrated in our approach as is described in Chapter 4.

Parameterization and Validation

After having implemented a parameterizable system for the DoG calculation, we performed parameter sweeps in order to estimate the total resource usage. The implementation is platform independent since the lower level modules are implemented in a behavioral manner.

The parameters that were swept are :

– Image width.

– Number of scales.
– Number of octaves.
– Maximum number of taps.

Note that the image height has no effect on the resource utilization due to the fact that we are performing stream processing on a neighborhood (kernel) as opposed to frame-based region.
A software floating point equivalent model was coded for comparison and validation purposes and the results were plotted in order to determine an appropriate value for the maximum number of taps. Note that the value will also be dependent on the base blurring standard deviation as shown in Chapter 4.

3.2.4 Huffman Encoding

After performing the DoG calculation, interest points coordinates are extracted as the local extrema. After that the keypoint description has to be calculated from the Gaussian image pyramid. The keypoint description involves floating point calculations which are often better offloaded to a host machine or dedicated processor.

We noticed two things, first of all blurred images tend to have very few high frequency components in the x and y directions and thus the histogram of a causal spatial differential image would tend to have a very steep peak around zero. Such images are well compressible using a Huffman encoder.

Figure 3.7 shows a test image and a Gaussian blurred version of it with their respective histograms. Notice that in this case the histograms are rather spread over the whole bit range.

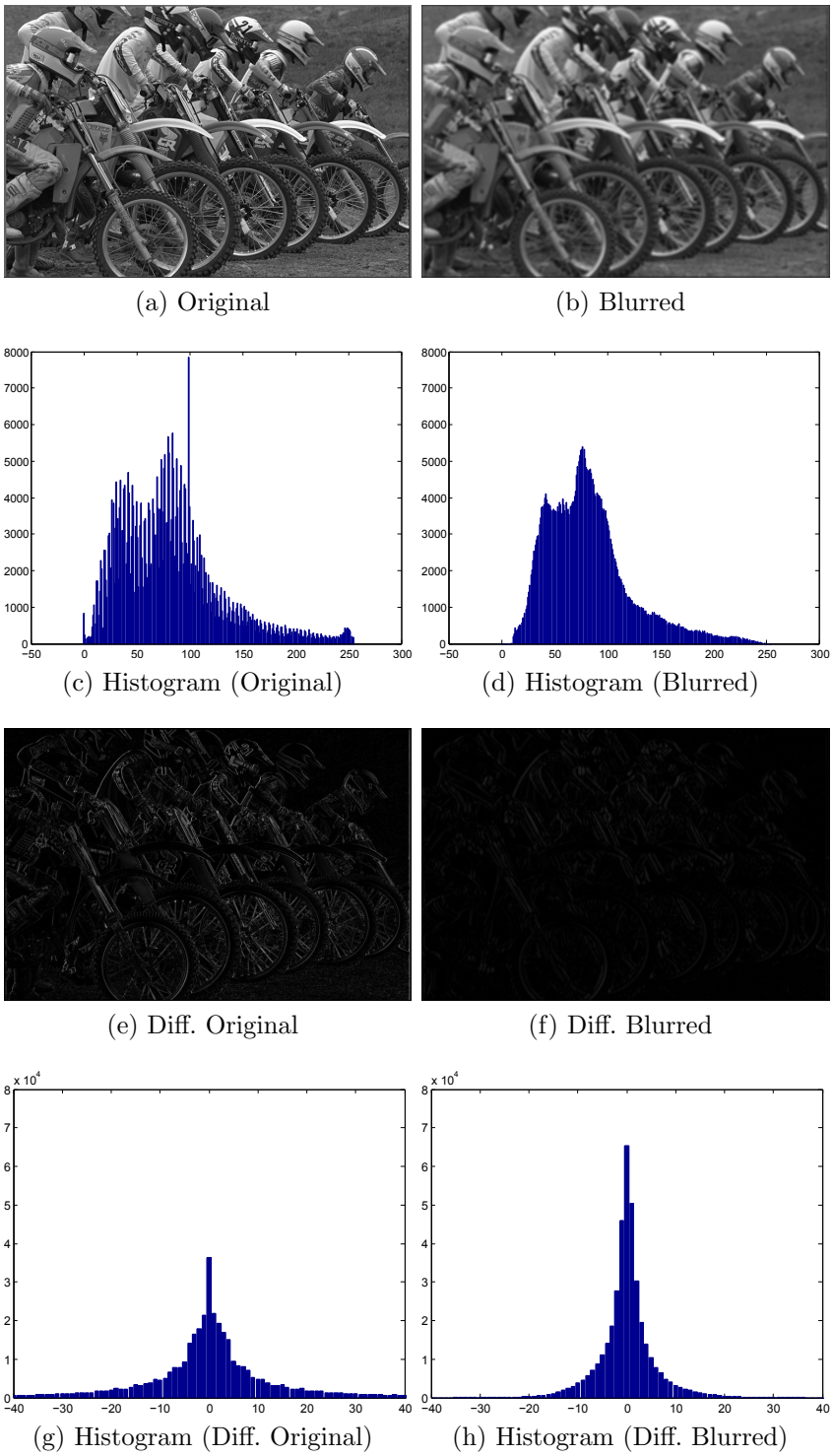
On the other hand, Figures 3.7e and 3.7f show what happens when the image is differentiated. Notice how the differential histogram (Figures 3.7g and 3.7h is much narrower than the original and also how the blurring has further gathered the majority of the pixel values around 0.

The second thing that we noticed is that since the same image is being blurred over and over to form new scales, the relative pixel variations from one scale to another are also minimal and we can thus take advantage of a causal scale differentiator prior to encoding.

In order to incorporate the best of the two worlds, we came up with a modified Paeth predictor as discussed in Chapter 4. The causal predictor takes into account spatial as well as scale variations to calculate a prediction value.

The parallel Huffman encoder that is proposed in Chapter 4 is able to process several pixels in parallel and can adequately be used to compress individual images as well as full image pyramids.

Such a method of compression would alleviate bandwidth bottlenecks without taxing the



(a) Original (b) Blurred

(c) Histogram (Original) (d) Histogram (Blurred)

(e) Diff. Original (f) Diff. Blurred

(g) Histogram (Diff. Original) (h) Histogram (Diff. Blurred)

Figure 3.7 Gaussian blurring and differentiation (Diff.) effect on the image histogram.

accuracy due to its lossless nature.

A simple flow of how to use our parallel Huffman encoder is shown in Figure 3.8. First of

all a training image (or set of images) is used to construct our fixed Huffman table ; this part is calculated in software based on the image histogram of the differential of these images, alternatively one could assume a fixed histogram distribution such as a Gaussian with a small standard deviation. Afterwards the calculated table is stored in BRAM and will be used for all the incoming images.

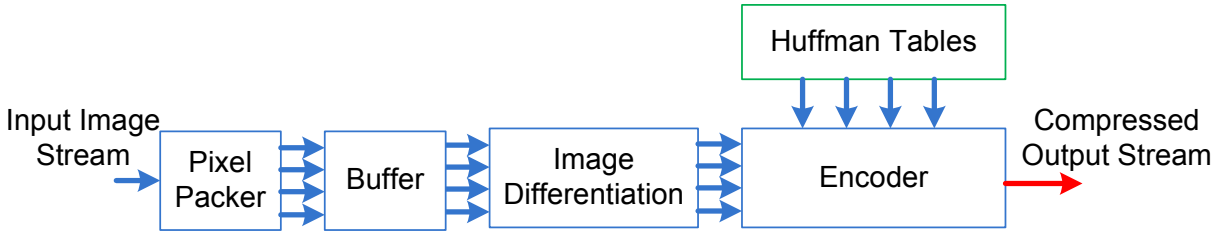


Figure 3.8 Example usage of Huffman encoder.

Note that even though we are using a fixed table that does is not re-calculated for every incoming frame, the compression results are comparable due to the differentiation stage that precedes the encoding. Afterwards every incoming frame is packed into a higher parallelism and buffered. Then the image is differentiated and fed to the Huffman encoder. The encoder will wait until enough bits have been accumulated and then a valid output packet is generated. Note that the encoder may temporarily output more bits than the input bandwidth (this occurs when an image contains a lot of edges), but the buffer will even out these variations in order to achieve and overall per-frame bandwidth reduction. The increased parallelism allows the encoder to "catch-up" whenever such edges are encountered without having to stall the input source.

3.3 Phosphene Map Estimation

A different path that was explored concerns the phosphene map estimation. Typically after performing object recognition a representation of this information will have to be conveyed to the patient. The representation can be represented as sounds queues or even as intra-cortical stimulations for the case of the Cortivision project.

A major assumption is that the phosphene map is already known for such a setup to work. An approximate map of the brain where the electrode placement would result in specific phosphene being illuminated in the field of view (FOV) has been presented in Dobelle and Mladejovsky (1974). Such a map is typically denser towards the center of the FOV and one should not expect to obtain a perfectly uniform grid.

Typically the phosphene map can be calculated based on the patient's feedback in order to be able to provide an error-minimized estimated map. Note that if the map is distorted, so will the stimulated phosphene image as illustrated in Figure 3.9.

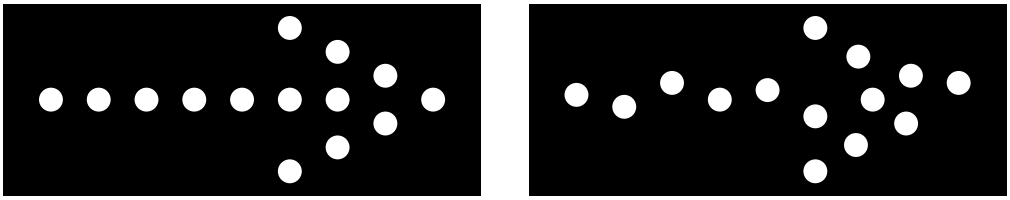


Figure 3.9 Original arrow symbol (left), distorted symbol (right) due to errors in phosphene map estimation.

A prototype that targets this issue and proposes an alternative to the ones found in the literature is presented in Chapter 4.

CHAPTER 4

AN IMAGE PROCESSING SYSTEM DEDICATED TO A VISUAL INTRA-CORTICAL STIMULATOR

Anthony Ghannoum, Ebrahim Ghafar-Zadeh, and Mohamad Sawan
Polystim Neurotechnologies Laboratory, École Polytechnique de Montréal
Submitted on 20 November 2012 to IET Image Processing

Abstract

In this article we focus on intra-cortical electrical microstimulation in order to cover the broader spectrum of visual impairment, with the aim of providing a better suited navigational and recognition aid for the patients. We present an overall modular architecture and focus on creating re-usable image processing tools that can be used for image simplification and recognition tasks. One of the main challenges in the image processing path is the real-time restriction ; hence we resort to FPGA hardware acceleration. Herein we demonstrate and describe the architecture of an image feature extractor based on the difference of Gaussians that is at once accurate, generic, and low on resources. This architecture also features a Huffman encoding engine that proves useful when resorting to SW-HW hybrid implementations, and a technique of calibrating and calculating the phosphene map.

4.1 Introduction

Visual impairment, defined as the full or partial loss of vision, is still not medically curable. Biomedical approaches have risen to address the issue ; these take advantage of the fact that electrical stimulations can be used to induce spots of light in the visual field called "phosphenes".

Different stimulation methodologies such as retinal implants and optic nerve stimulation are currently being employed Humayun *et al.* (2003); Sakaguchi *et al.* (2009). Our group targets the broader spectrum by using an intra-cortical stimulator which is made of a camera, an image processing module, a wireless transmitter, and the stimulating electrodes. The direct stimulation of the visual cortex bypasses the eyes and optic nerves thus these do not need to be intact as compared to the other methodologies.

Fig. 4.1 shows an overview of how the system works. Basically, the head is mounted with a frontal camera which might need to be high speed or even perform some minor processing

Ghannoum and Sawan (2007). What follows is an image processing module responsible for "simplifying" the image content and performing recognition tasks. After that, the processed image has to be fitted into the existing phosphene map in a way that is easy for the patient to interpret. In order to avoid infections and other complications, a wireless transmitter is then used to transfer the stimulation parameters and power across the skin. Finally, a set of microelectrode matrices will perform the actual current injection, hence stimulating the visual cortex and forming an image in the patient's visual field.

One should note that the phosphene map or actual phosphene location is static ; it depends on the electrode placement. Therefore, the map is expected to be an irregular one and will have to be calibrated with the patient's feedback as will be seen later on in Section 4.4.

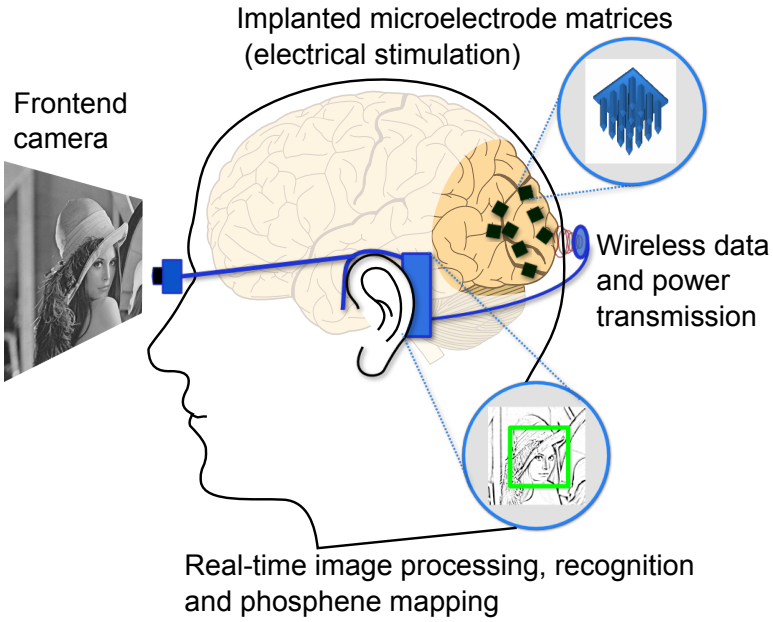


Figure 4.1 Overview of the Cortivision system.

Typically the number of stimulation sites is limited, hence the need for image pre-processing prior to stimulation to extract the salient information and project it on a reduced map/image than can be perceived and interpreted by the patient. Image segmentation, edge detection Buffoni *et al.* (2005), and stereo processing techniques for building disparity maps, have been used to aid in navigation. However, these methods do not permit actual object recognition, sought-after in this paper, which calls for more complex processing to adequately extract the information prior to stimulation. The Scale Invariant Feature Transform (SIFT) algorithm Lowe (2004) presented itself as an ideal candidate for the purpose, and is extensively used in the literature for different applications such as feature matching, Simultaneous Localization and Mapping (SLAM), stereo vision, and image mosaicing. It has been shown

to be one of the most stable feature detectors Mikolajczyk and Schmid (2005) compared to other approaches such as Speeded-Up Robust Features (SURF) Bay *et al.* (2006) which relies on integral images for quick approximate calculations.

In terms of speed, the downside of the software algorithm lies in the calculation complexity of the feature localization stage which involves an image pyramid (Gaussian Scale Space) calculation as the one shown in Fig. 4.2, which is performed by consecutively blurring an image with a Gaussian kernel for several scales and then sub-sampling the results to form another octave. Afterwards, the Difference of Gaussians (DoG) is used to find extrema points that are stable across scales. Hence, hardware implementations and accelerators have increased in popularity Bonato *et al.* (2008); Chang and Hernández-Palancar (2009).

The main drawbacks of the hardware, FPGA-based, architectures found in the literature are typically the precision and resource usage ; therefore such implementations would tend to simplify the algorithm to fit the specific application needs to minimize resources for a particular platform.

Moreover, when an FPGA is used as a pre-processor or when inter-frame dependent processing is considered, data transfer bandwidth can become the main bottleneck of the application. To overcome that limitation, compression schemes such as JPEG, and MPEG have been introduced to reduce and relax such limitations. Yet, some medical applications cannot afford any precision loss induced by lossy encoders.

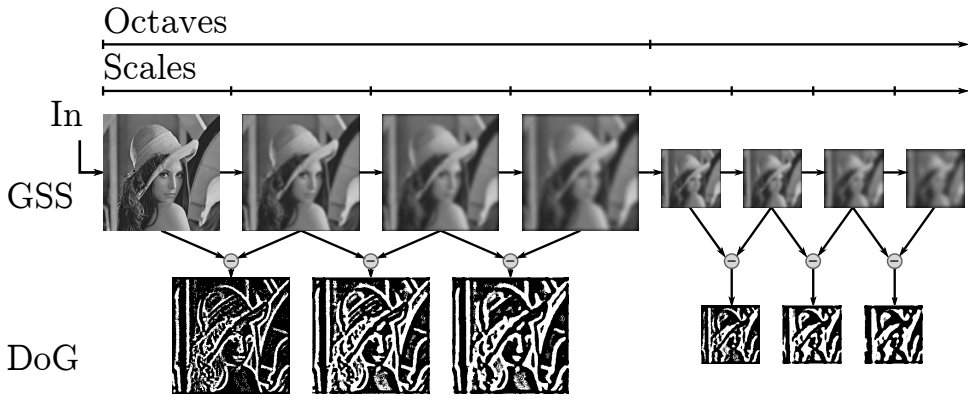


Figure 4.2 SIFT : Image Pyramid. 2 Octaves 4 Scales. Blurring at every scale and subsampling (2×2) at every octave. Notation : Gaussian Scale Space (GSS) and Difference of Gaussians (DoG).

Lossless compression methods such as Run Length Encoding (RLE) require the image to have long runs of similar pixels, to achieve good compression. Alternatively, entropy encoders, such as arithmetic and range coding, can be used to encode symbols that occur probabilistically more frequently in less number of bits thus reducing the overall number of transmitted

bits. The former's major downside is the use of floating point operations which makes it troublesome for an FPGA implementation. The latter, uses integer numbers for its representation but division operations that are involved in the algorithm and re-normalizations of the range by using power operators would complicate such a design even more if input parallelization (multiple pixel input) was to be considered.

Noting the inherent data dependency in the Gaussian Scale Space (GSS) pyramid, we propose an architecture for a pipelined parallel Huffman encoder that can be applied to general lossless image compression to reduce bandwidth requirements. We implemented a generic interleaved VHDL image pyramid that makes use of the sub-kernel trick, to save resources, and incorporated with our encoder. The proposed DoG pyramid is portable and parameterizable in image width, scales, octaves, and maximum number of filter taps, which allows tailoring resource usage according to application needs. The presented results verify that the FPGA implementation outperforms software ones, allowing real-time processing. The image

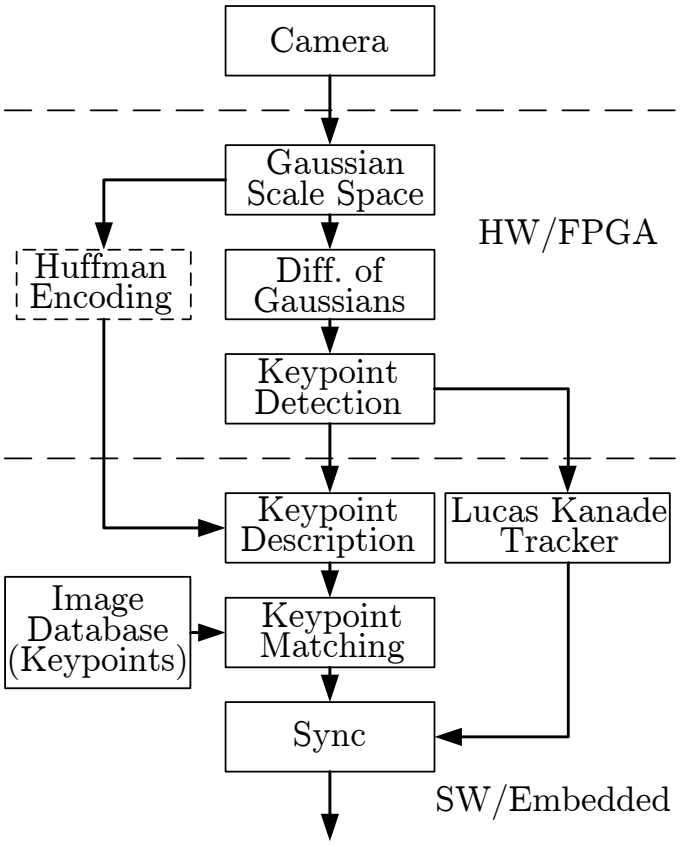


Figure 4.3 Image processing system used for object recognition and tracking

processing architecture that we present is illustrated in Fig. 4.3. It is a hybrid implementation where firstly the Gaussian scale space is calculated, followed by the difference of the adjacent

scales, and an inter-scale non-maxima suppression which performs the keypoint localization. It should be noted that this part was chosen to be implemented on an FPGA due to the fact that it is the most calculation intensive part and is highly parallelizable and would be required for real-time processing.

Moreover the Huffman module compresses the intermediate blurred images and transfers them to the next processing stage.

The software/embedded part of the system is responsible for matching the keypoints to a pre-built database. The matching is performed using the Random Sample Consensus (RANSAC) to eliminate false matches, and accelerated using k-d trees as in Lowe (2004). Moreover, in order to be able to track an object for the case where the object might be "lost" from one frame to the next, the Lucas-Kanade optical flow tracker is run as a separate thread hence providing continuous tracking.

The remainder of the paper is organized as follows, we discuss in Section 4.2 the new differential method of Gaussians architecture, Section 4.3 shows the parallel Huffman encoder and the proposed pipelined architecture, Section 4.4 presents a phosphene calibration technique which is a work in progress, Section 4.5 shows the synthesis results for a parametric sweep and the compression ratios of the Huffman encoder, and the conclusion is in Section 4.6.

4.2 Difference of Gaussians

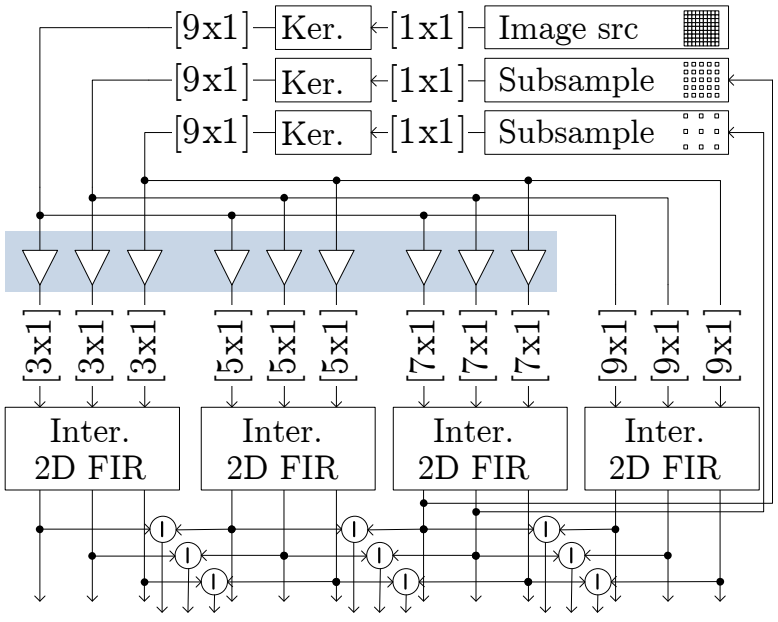


Figure 4.4 Interleaved DoG. (3 Octaves, 4 Scales) Notation : Kernelizer (Ker.) and Interleaved (Inter.)

The repetitive Gaussian blurring may be time-consuming for a host computer especially if the kernel sizes tend to be large ; FPGAs on the other hand are ideal candidates for such types of processing that can be pipelined. Image processing on FPGAs can be roughly divided into three main types :

- Point operations (e.g. color conversion, gain and offset)
- Neighborhood operations (e.g. filtering, edge detection and morphology)
- Frame-based operations (e.g. frame buffer, frame difference and affine transforms)

Point operations operate on the incoming pixel stream. Neighborhood operations would typically buffer some lines of the image in order to provide a kernel to work with, i.e. the region around the current pixel (borders are typically handled by constant padding, mirror, or replication). In order to do that, the FPGA internal Block RAM (BRAM) is employed as dual ported cyclic RAM buffers. Alternatively for relatively short lines, one could use the Xilinx dedicated shift register blocks SRL16. Finally, frame based operations require buffering at least a whole image frame to be used at a later time. For small images, the internal FPGA BRAM would prove to be sufficient for such a task, but for larger images, one would have to resort to external RAM such as DDR SDRAM, RLDRAM, or SRAM. Note that when resorting to use DDR RAM, one has to keep in mind that random accesses such as image rotation (i.e. accesses that induce page changes) would highly penalize the effective bandwidth of the memory.

For the current architecture, we use neighborhood operations while effectively utilizing our BRAM resources. As already mentioned, a pipelined interleaved architecture for the DoG calculation was incorporated in Chang and Hernández-Palancar (2009).

It is to be noted that the DoG implementation in Bonato *et al.* (2008) utilizes $O * S$ convolution blocks hence would be wasteful in terms of FPGA resources.

The fact that the image is sub-sampled (data size is divided by 4) at every octave induces a lot of idle cycles for the higher octaves, thus allowing us to interleave the processing. The assumption is that the pixel rate is half the FPGA clock rate, otherwise the first octave will have a dedicated processing path, and the remaining octaves can be interleaved. Note that, the Gaussian kernel is a separable one, and hence the filtering operation can be performed in the vertical direction, followed by the horizontal direction, thus saving some multiplier resources. The approach in Chang and Hernández-Palancar (2009) uses an interleaved cascade of filters of fixed size e.g. 7×7 to obtain the intermediate Gaussian images. What this means is that for every scale at every octave, the image has to be buffered repetitively for the new filter cascade to be implemented ; Although this is an improvement over the architecture presented in Bonato *et al.* (2008), it still consumes the internal BRAM resources.

The Gaussian standard deviation σ_{cn} for every cascaded scale n can be calculated as such (for 6 scales) :

$$\sigma_{cn} = \sigma_o 2^{\frac{n-1}{3}} \sqrt{2^{\frac{2}{3}} - 1}$$

Where σ_o is the base blurring of the pyramid which is typically set to either 0.8 or 1.6. Typically the Gaussian kernel size is chosen to be $4 * \sigma$ or $6 * \sigma$. What we propose is the use of the sub-kernel trick instead of cascading filters, hence the standard deviation for the scales is non-cascaded can be calculated as follows :

$$\sigma_n = \sigma_o 2^{\frac{n}{3}}$$

By calculating the worst case σ, found at the top scale, we can set an upper limit for the maximum kernel size. Fig. 4.4 shows an overview of the proposed DoG architecture for 3 octaves, 4 scales, and a maximum kernel size of 9.

The "Kernelizer" module represents a pixel buffering block, and the line notation $[M \times N]$ suggests that the output of the current block is a buffered kernel stream (instead of a $[1 \times 1]$ pixel stream). Note that if M = 1, the block performs a purely horizontal buffering, and hence no BRAM is used, otherwise internal line buffers are required.

The shaded triangular connections, shown in Fig. 4.4, represent the Sub-Kernel operation ; this block does not cost anything in hardware since we would be basically selecting a subset of the kernel stream to form a smaller kernel stream at its output. The sub-kernel is typically centered. It should be noted that although we took advantage of the Gaussian separability, we adequately placed the vertical kernel first, hence sharing BRAM resources for the different scales. The sub-sample blocks remove every other pixel and every other line. The Interleaved 2D FIR block is shown in Fig. 4.5. It is made up of 2 Finite Impulse Response

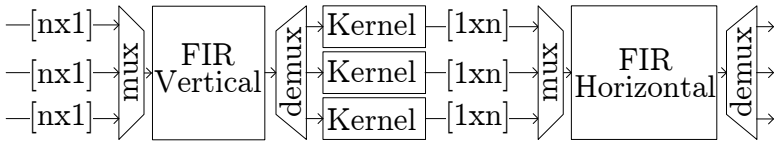


Figure 4.5 2D Interleaved FIR filter. Taking advantage of the of 2D Gaussian separability, we save resources by performing a 1D vertical filtering followed by a 1D horizontal one.

(FIR) filters for vertical and horizontal filtering, and other mux/demux blocks to perform the interleaving operation. The internal architecture of the FIR filters is typically platform dependent. For our current system, we are using the architecture shown in Fig. 4.6. Due to

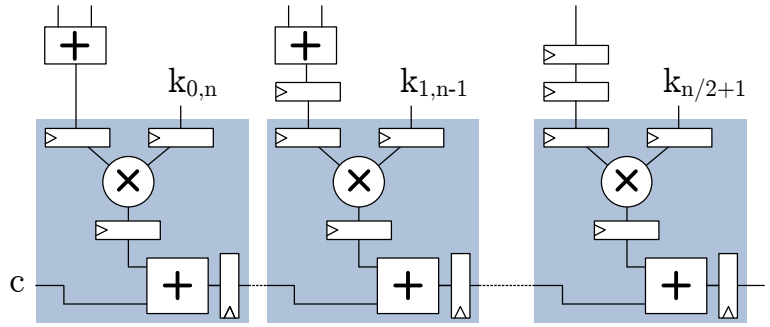


Figure 4.6 Symmetric FIR with pre-adder. Notation : Registers are indicated with a triangular notch.

the symmetric nature of the Gaussian kernel, we use a pre-adder to reduce the number of required multipliers. Also, in order to get a good fit inside the Xilinx DSP48 components we used an adder chain instead of an adder tree architecture. We can also get a free adder for the first stage where we can cascade the "c" input shown in Fig. 4.6 ; this enables us to perform rounding operations (add-half truncate).

One should keep in mind that FPGA devices are increasingly becoming equipped with dedicated multipliers or multiply accumulate MACC engines ; such components are also able to work at much higher clock rates than the rest of the FPGA fabric. The advantage of the proposed architecture is twofold : we save BRAM resources and the numerical accuracy is more precise compared to the cascaded filters.

BRAM resources are reduced in two ways : first due to the fact that a single large kernel is used for all the scales in an octave, instead of a separate smaller kernel for each octave. Second, when the sub-kernel trick is used, the output stream of all the sub-kernels are synchronized, and hence at the subtraction stage, shown at the bottom of Fig. 4.4, no further synchronization is required, on the other hand, the cascaded architecture would induce several lines of delay at every cascade stage, and hence the pixel streams would need to be resynchronized prior to the final differentiation.

4.3 Parallel Huffman Encoder

For our purposes the Huffman algorithm presented itself as an ideal candidate. Huffman encoding, is a way of distributing the number of bits for each symbol based on the probability distribution. The encoder can be divided into three categories that store the encoding tables

differently.

- Static tables : typical probabilities are used to build a fixed table that would be used for both the encoder and decoder.
- Adaptive tables : the table is updated on the fly while the transmission is occurring. Note that in such a case, the decoder will also have to update its own table.
- Two-pass encoding : The probability distribution of each frame is calculated on the first pass and the resulting table is sent to the decoder as a header along with the frame which is encoded on the second pass.

We present a parallelizable architecture for a Huffman Encoder. For our application, we opted for a static table implementation, where no preceding table header is required. The same architecture would work with adaptive tables requiring more logic to update the tables.

4.3.1 Image Differentiation

Image differentiation is required to center the image histogram around 0, and can be performed as inter or intra-frame. The former would perform extremely well for a fixed camera layout, while the latter would typically differentiate the image in a causal manner (left-to-right, top-to-bottom) enabling us to undo the differentiation on the receiving side.

For our specific application, of compressing an image pyramid made up of S scales and O octaves formed under the GSS, a modified version of the Paeth predictor Paeth (1991) (used in the PNG and TIFF standard formats) was used to obtain our differential images. The DoG, inter-scale difference, is included in our differentiator since the pyramid scale images are very similar to each other.

Let a, b, c, and d represent the respective left, top, top-left, and lower-scale pixels. The original Paeth predictor approximates the slope of the pixel's surrounding region using the "poor man's Laplacian" $G(a, b, c) = a + b - 2c$ and produces an initial estimate P by adding the slope to c. The pixel closest to the initial estimate is hence used as the predicted value.

Our modified predictor takes into consideration the scale difference and reverts to the original Paeth predictor definition if this difference is too large. The actual encoded value is the difference between the current pixel and the predicted value. This predictor is useful for the base image which has no scale below it, and when excessive blurring cause adjacent

pixels to have closer values to each other than to the scale below.

$$
\begin{aligned}
&P := a + b - c;\\
&Pa := abs(P - a); \quad Pb := abs(P - b);\\
&Pc := abs(P - c); \quad Pd := abs(P - d);\\
&if\ (Pa \le Pb)\ \&\ (Pa \le Pc)\ \&\ (Pa \le Pd)\ \ return\ a;\\
&else\ if\ (Pb \le Pc)\ \&\ (Pb \le Pd)\ \ return\ b;\\
&else\ if\ (Pc \le Pd)\ \ return\ c;\\
&else\ return\ d;
\end{aligned}
$$

Fig. 4.7 shows an overview of the proposed architecture for input parallelism of 4. The Huffman table is stored in a ROM for static tables. After that a pipelined Huffman concatenator shifts the parallel inputs by the appropriate number of bits and appends the results. The final stage, the Huffman barrel shift register buffer, accumulates enough bits prior to outputting a valid, fixed-size, compressed packet.

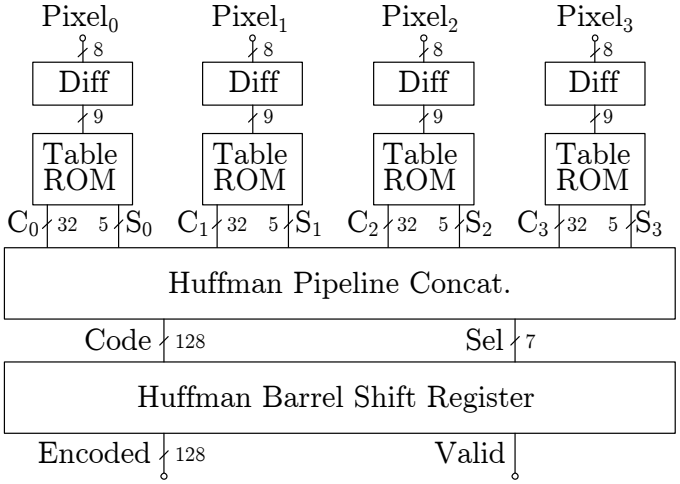


Figure 4.7 Simplified Block Diagram of Parallel Huffman Encoder

4.3.2 Architecture of the Proposed Encoder

Barrel Shifter

The data coming from the ROM is made up of pairs of *code* bits and a *select* value. The *select* value indicates how many of the bus bits are valid and should be shifted in the final buffer. In order to perform variable shifts, a barrel shifter is used. A simple and effective

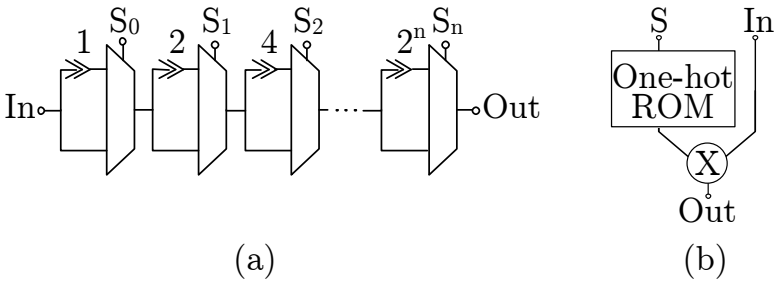


Figure 4.8 Barrel Shifter (a) Basic Architecture (b) Multiplier-based *(Notation : ≫ logical bit shift)*

architecture such as the one shown in Fig. 4.8 (a) is used. Note that the *select* bits define the maximum combinatorial path, which is acceptable in most cases. If timing closure is not reached due to that path, the barrel shifter can further be pipelined.

An alternative architecture can be implemented by using the large, high-speed embedded multipliers that are found on the newer FPGA devices (like the Xilinx XtremeDSP DSP48 in the Virtex families). As shown in Fig. 4.8 (b), by coupling a one-hot encoded ROM with a multiplier, one could implement a high speed barrel shifter.

Encoder

In order to support parallel inputs, we used the pipelined Huffman concatenator architecture shown in the top part of Fig. 4.9. The *selects* S_n of the respective *code* C_n is used to shift the next value C_{n+1} and then the data is *OR-ed* to perform the actual concatenation. The Huffman table that contains the *codes* is assumed to be zero padded for the unused bits in the *code*. In order to cascade that, the *selects* are added and the result is cascaded to the next barrel shifter of increasingly larger size.

The resulting pair is then fed into the Huffman barrel shift register shown in the bottom part of Fig. 4.9. This module is the actual buffer and output control unit of the encoder. It takes a single *code/select* pair as input and buffers the *codes* until a sufficient number of bits have been accumulated, after-which a valid signal is emitted and the buffer feedback is shifted right to align the data.

4.4 Phosphene Map Calibration

An important factor, that impacts the phosphene "image" that is perceived by the patient, is the actual phosphene map calibration. The phosphene map can be thought of as the pixels that the patient will perceive. These are not regularly distributed, and their locations are determined by the actual electrode placement in the visual cortex. The doctors

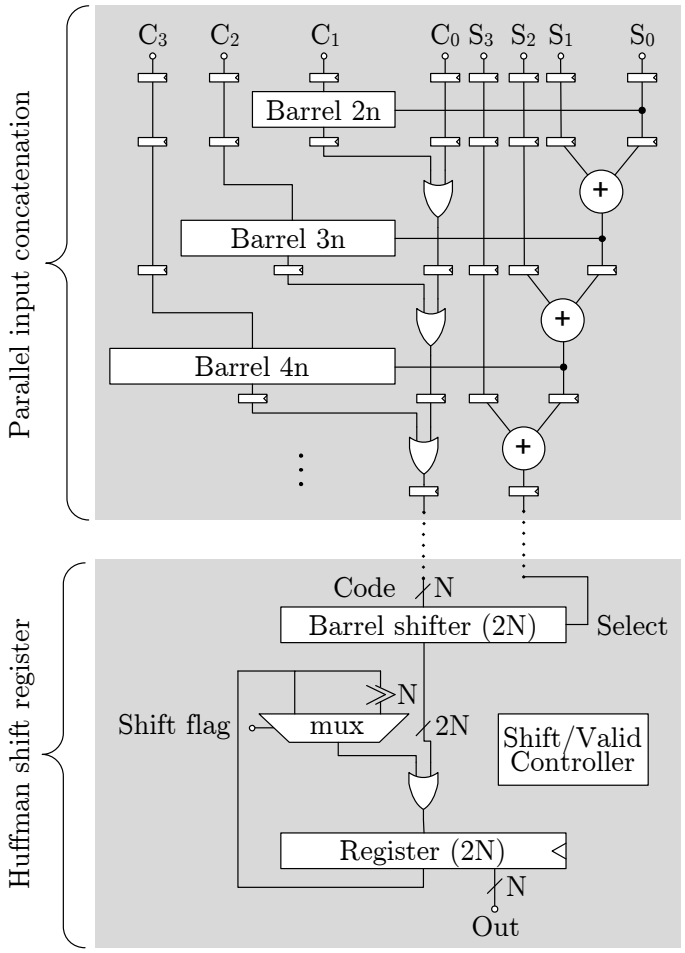


Figure 4.9 Parallel Huffman Barrel Shift Register

typically have a map of the brain Dobelle and Mladejovsky (1974) that would indicate the approximate electrode sites, and the electrodes are built as stimulation matrices, hence fine adjustments can be performed. Various methods exist in the literature to calibrate and adequately approximate the map. The most common is a set-up with a tactile screen Chai *et al.* (2008) or touch screen LCD ; the patient would be presented with two points (illuminated phosphene pairs) at a time, and would have to indicate, using the touch screen, the distance between the two perceived points. After collecting an adequate number of pairs samples, an error minimization technique can be used to approximate the overall phosphene map.

The previous set-up can be quite limited, first by the touch resolution and second by the number of pairs that can presented. What we propose is a set-up similar to the one shown in Fig. 4.10. We basically use fiducial markers on a transparent table, these markers are identified and tracked using a high resolution camera. The image processing was performed

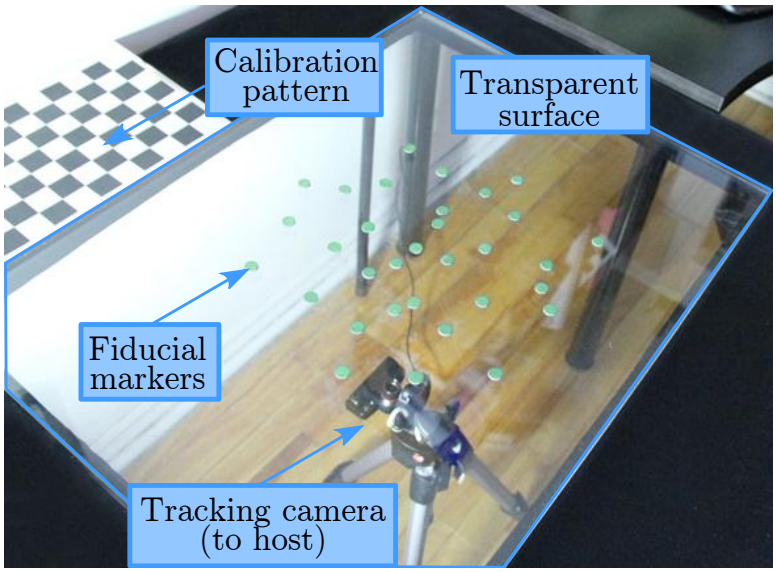


Figure 4.10 Set-up of the phosphene map calibration.

using the OpenCV Library Bradski (2000). First the camera's intrinsic distortion matrices (fish-eye and barrel distortions) are calculated. After that, the camera is placed as accurately as possible, and a checker-board grid is used to perform a bird's eye rectification.

The current fiducial markers that we are using are round, green colored markers that can be easily identified in the HSV color space. A 2D block-based non-maxima suppression Neubeck and Van Gool (2006) is used to find the markers, and we are able to track up to 256 markers in real-time. The tracking uses the global nearest neighbor scheme Yilmaz *et al.* (2006) based on the Jonker-Volgenant algorithm LAPJV Jonker and Volgenant (1987).

This method offers several advantages over existing methods. For instance, fiducial markers can be located with sub-pixel accuracy or a large number of markers can be used at once. Furthermore, tracking can be used to update the patient's image in real-time.

Various options are currently being explored. Presenting the patient with different constellations of phosphenes instead of pairs of them might increase the calibration accuracy during the triangulation and error minimization stage. Another alternative would be to present the patient with a full phosphene image of simple geometrical objects (with all the phosphene/markers at once), the patient will then have to adjust the markers' map in real-time to get an image that is not distorted. Also symbols and objects can be modulate as pulsating objects and/or swept across the visual spectrum. Further testing will be performed with volunteers with head-mounted displays to evaluate the accuracy of the proposed method. This work is currently in progress in the goal of achieving a more accurate phosphene map estimate, which in turn would lead to high precision stimulation images that are perceived

with minimal distortions.

4.5 Simulation and Experimental Results

A generic parameterizable VHDL implementation of the Difference of Gaussians and Huffman encoder was implemented on a Xilinx ML605 Virtex-6 board and tested as a black box, in a hardware co-simulation loop in System Generator. The architecture can run at 100 MHz while returning a result at every two clock cycles (due to the interleaving), and could run up to two orders of magnitude faster than a software implementation Vedaldi and Fulkerson (2010) for images of 640×480 running on a host machine (Intel Core i5 @ 2.27 GHz, 6 GB RAM).

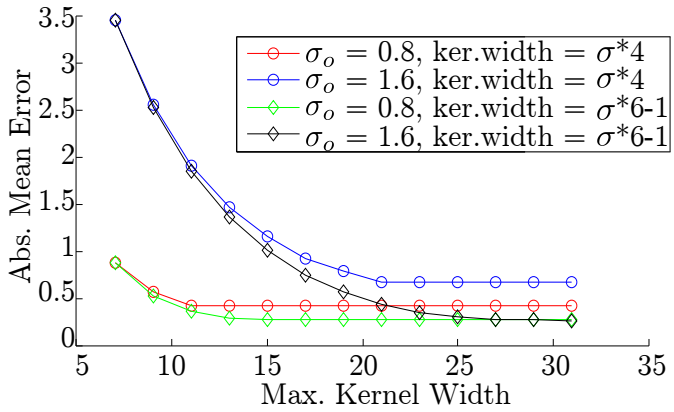


Figure 4.11 DoG absolute mean error, compared to a double precision 31-tap kernel (for all octave/scale combinations).

4.5.1 DoG Precision

In order to evaluate the numerical precision of the employed architecture, it was compared to a double precision software model using filter of 31-taps (borders errors are ignored). Fig. 4.11 shows the absolute mean error plot for Gaussian pyramid filtering. The base Gaussian standard deviation is shown for two values, $\sigma_o = \{0.8, 1.6\}$. Also the kernel width, or sub-kernel width, at each scale is chosen according as a function of σ. A limiting parameter is set on the maximum kernel width (this will affect the number of BRAM resources, since it defines the upper limit on the vertical kernel width prior to the sub-kernel operations). Table 4.1, shows the precision of the cascaded difference of Gaussians implementation Chang and Hernández-Palancar (2009). Using our implementation, a maximum kernel size of 15 can be used to obtain better precision and for comparison a kernel size of 31 is also shown. The

precision was measured in terms of the absolute mean error, the standard deviation of the error, and the maximum bits per pixel (bpp) error.

4.5.2 VHDL Synthesis

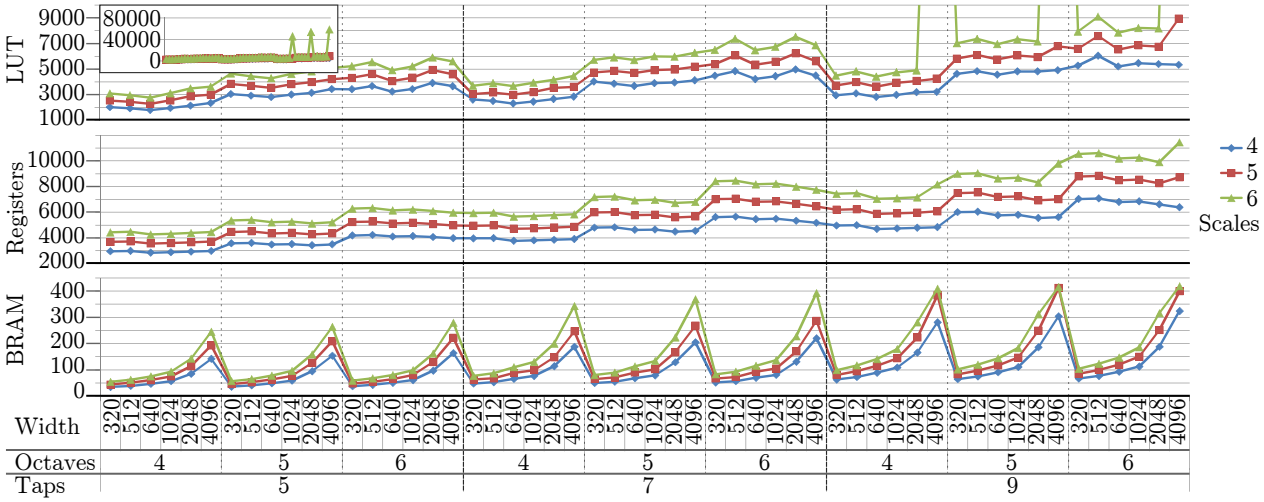


Figure 4.12 Cascaded DoG Resource Usage – Parametric Sweep

The design was implemented in portable VHDL code by inference, no Xilinx specific components were used. Fig. 4.12 shows a synthesis parametric sweep for the Cascaded DoG module.

Table 4.2 shows the resource usage of the $O * S$ cascaded implementation Bonato *et al.* (2008), the interleaved cascaded architecture presented in Chang and Hernández-Palancar (2009) on a Virtex-2 platform. For comparison purposes, we have also implemented the interleaved cascaded architecture on a Virtex-6 platform (shown in the third column) ; and finally our new sub-kernel based architecture is shown in the fourth column. The employed parameters are : image width of 320 pixels, 4 octaves, and 6 scales. The cascaded architecture uses 7 tap Gaussian filters and for our sub-kernel architecture a maximum kernel width of 15 was chosen. From that, we are able to deduce the resource usage savings that are gained by our architecture especially for the BRAM, which further simplifies the placement and routing stage.

It should be noted that for the Virtex-2 and Virtex-6, the main differences can be categorized as 18kb BRAM/4-input LUTs and 36kb BRAM/6-input LUTs respectively. Our implementation of the interleave cascaded architecture is capable of flow control, border handling, and flushing at the end of frames without requiring prior image buffering which would justify

Tableau 4.1 Precision of cascaded DoG and sub-kernel DoG

	Cascaded FIR		Sub-Kernel (15)		Sub-Kernel (31)	
σ_o	0.8	1.6	0.8	1.6	0.8	1.6
Absolute mean error	0.3409	1.2200	0.2760	1.0041	0.2760	0.2640
Standard deviation error	0.2549	1.6678	0.1825	1.5665	0.1825	0.1664
Maximum bpp error	2	19	2	19	2	1

the slight increase in BRAM. It can be observed that the LUT usage increases with taps, octaves, and scales. As for the width dependency, we may notice a slight curvature. This could be explained by the fact that, in the image pyramid, images are sub-sampled at subsequent octaves. For small images, this sub-sampling reduces the image width to an extent where the required line buffers are more readily implemented or inferred as LUTs rather than BRAM components. Current Xilinx devices can configure their LUTs as SRL16E components (Shift Register Look-Up-Table) which can be well cascaded to perform small line buffering functions and in some cases reduce resource usage. From Fig.4.12, we can also notice an explosion of LUT usage for the high scale, tap, and width combination. The reason behind that is simply because we ran out of BRAM resources on our device and the synthesizer reverted to using LUTs ; such cases are not practically implementable. The register usage is quite independent of the image width, but on the other hand the BRAM usage is directly dependent on the width. As shown in Fig. 4.12, the architecture consumes a lot of BRAM resources for large images and hence trade-offs must be made in case the large images are to be processed.

Tableau 4.2 DoG Resource Usage

	$O * S$ Cascade Stratix II †	Interleaved Cascade Virtex-2 ‡	Interleaved Cascade Virtex-6 (Compare)	Sub-Kernel Virtex-6 (This work)
LUT	15137	6880	3702	3669
Registers	7256	6028	5942	3193
BRAM	0.91Mb	120 (2.1Mb)	77 (2.7Mb)	21 (0.74Mb)

†Bonato *et al.* (2008)

‡Chang and Hernández-Palancar (2009)

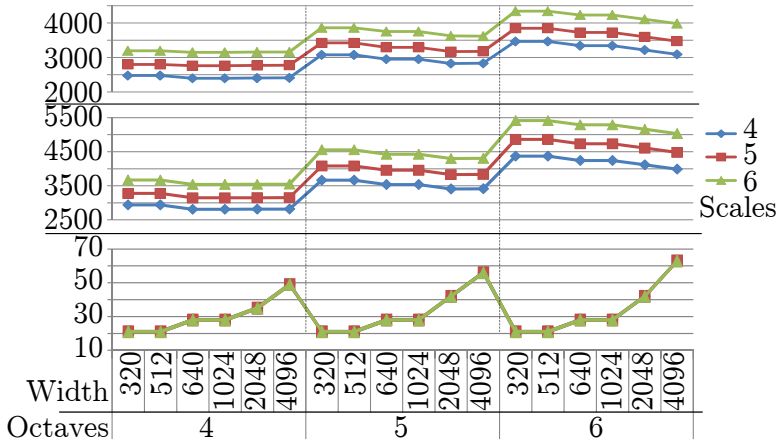


Figure 4.13 Sub-Kernel DoG Resource Usage – Parametric Sweep

On the other hand, our implementation using the sub-kernel trick is shown in Fig.4.13. We may notice that our architecture's BRAM usage is lower that the cascaded one and is independent of the scale configuration used.

As for the Huffman encoder, the presented architecture was synthesized for various parallelisms (simultaneous pixels or scales) and produces the resource usage shown in Table 4.3. Note that BRAM resources can be halved (for even parallelism) by sharing Huffman tables as true dual port RAM. Depending on the image pixel parallelism and the memory buffer architecture that follows, one might opt for a single interleaved encoder with high parallelism, as we did, or several encoders with a more elaborate control.

4.5.3 Huffman Encoding

The Caltech-256 image dataset was used to test the compression ratios of the Huffman encoder. A fixed Huffman table was built by taking 10 training images from each object category. The same table was used to encode all the images and their pyramid octave/scale combinations. Table 4.4 shows the compression ratio achieved for different parameters. Note that the parallelism parameter stands for the number of simultaneous pixels n that are input at every clock cycle to the compression core as shown in Fig. 4.14. This is required since, in

Tableau 4.3 Parallel Huffman encoder resource usage

Parallelism	1	2	4	6	8
LUT	152	393	1169	2182	3451
Registers	254	708	1737	3741	5947
BRAM	2	3	6	9	12

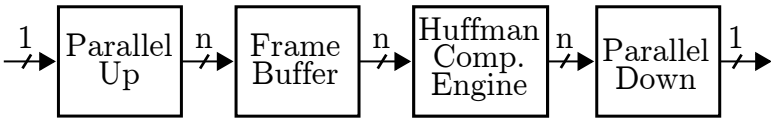


Figure 4.14 Compression Buffering Scheme

compression applications one would typically have an image buffer (or a multi-line buffer) followed by the compression engine. The buffer would decouple the incoming input stream from the compression core which should be able to support a higher input parallelism n than the uncompressed stream in order to compensate for noisy bursts.

Hence, when using a smaller number of scales, the blurring difference per scale increases thus making the original Paeth more attractive than the DoG for computing a pixel differential that is closer to zero. Fig. 4.15 shows samples of the minimum/maximum compressed categories. One can see that smoother images (mars, galaxy) are better compressed that the ones with a lot of edges (mussels, trilobite).

Figure 4.15 Caltech-256 samples : mars, galaxy, mussels & trilobite

4.6 Conclusion

We presented a new scalable architecture for the difference of Gaussians that makes use of the sub-kernel trick to save FPGA resources while still being more accurate than alternative implementations Chang and Hernández-Palancar (2009) ; the architecture shares line buffers (BRAM) across scales and does not require re-synchronizing the scales streams prior to differentiation. Also, our DoG implementation, which is used for feature extraction, is parameterizable as compared to the ones found in the literature Bonato *et al.* (2008)Chang and

Tableau 4.4 Huffman Encoder Compression Ratio

	Scale = 4		Scale = 6	
	$\sigma_0 = 0.8$	$\sigma_0 = 1.6$	$\sigma_0 = 0.8$	$\sigma_0 = 1.6$
Avg.%	29.68	25.83	29.22	24.56
Min.%	19.69	17.59	18.75	16.84
	mars	galaxy	mars	galaxy
Max.%	41.24	35.49	40.73	33.45
	mussels	mussels	trilobite	mussels

Hernández-Palancar (2009). This is essential for our application in that it gives us the required flexibility bearing in mind that the feature description and matching, object recognition stage that follows, might be iteratively changed or tweaked to fit the underlying hardware or precision requirements. A parametric sweep was presented that shows the resulting resource usage for different image widths, octaves, scales, and taps.

Moreover, the image pyramid that is reused in the subsequent processing stage was noted to be the main cause of transmission and memory bottlenecks, hence the parallel Huffman encoder architecture was presented and applied with a modified Paeth predictor which takes advantage of the inter-scale similarity. The encoder showed average lossless compression ratios of 27.3% on the Caltech-256 image dataset, hence alleviating the aforementioned bottlenecks. Finally, a work in progress of new method of phosphene map calibration was proposed. The system, being based on a camera set-up and image processing tools, is highly adjustable and is capable of real-time tracking of 256 fiducial markers. This allows us to experiment with larger phosphene constellations and even calibration techniques based on simple geometric shapes with real-time patient feedback.

Acknowledgment

The authors would like to acknowledge support from NSERC and the Canada Research Chair on Smart Medical Devices.

CHAPTER 5

GENERAL DISCUSSION

We discuss in this chapter how the article presented in Chapter 4 and the adopted methodology pertains to the literature review in Chapter 2.

5.1 Algorithm Robustness

One of the main targets was to use a robust algorithm that can perform object recognition in highly variable environments. Simplistic methods such as template matching that were mentioned in Chapter 2 proved to be very sensitive to even the smallest of image variations.

The multithreaded software implementation of the SIFT and Lucas-Kanade tracker was used as a robustness guide. We were able to prove that the system is able to recognize objects with variations in illumination, scale, rotation, affine transformations, and even perspective transformations up to a certain extent.

The SIFT, being a feature-based method, proved to be robust due to its loose modeling of an object using gradient histograms descriptions around scale extrema points (comparable to corner points). These features are at very discriminative while still allowing minor affine variations and illumination changes.

5.2 Hardware Implementation

The hardware (FPGA) implementation was performed in VHDL. We focused on maintaining well defined module interfaces that allow proper flow control to be propagated in case of bandwidth bottlenecks or processing stalls.

The modules were coded by inference, using behavioral modeling and following synthesis templates, in order to obtain portable and reusable code.

In the DoG implementation, we focused on obtaining a design which is, resource efficient, highly parameterizable to fit different application needs, and comparable in precision to a software floating point implementation.

The resource efficiency was mainly gained in terms of saving BRAM resources by the used the proposed *sub-kernel trick* and performing the vertical filtering prior to the horizontal one. This, coupled with the fact that the DoG system is also interleaved in octaves, proved to beat other implementations in the literature such as the cascaded approach Bonato *et al.* (2008) and the interleaved cascaded one Chang and Hernández-Palancar (2009).

An important thing to note, is that when we compared ourselves to the other architectures in the literature in Table 4.2 we were using rather small images (320×240) which does not clearly show the gains achieved by our proposed architecture. One might notice that the Stratix II implementation by Bonato *et al.* (2008) is quite close to ours in terms of BRAM usage although they are using a cascaded approach. There are two reasons for that, first of all they are treating one Octave less than we are and second they are using an Altera device that contains smaller memory BRAM components which are better suited for small images whereas we are always rounding up to the nearest 18kbit BRAM at every time.

A graph that explains the BRAM scaling with image size is shown in Figure 5.1. On the left, we can see the resource usage if we do not need to perform rounding to the nearest BRAM size, whereas on the right we show how the usage scales with the rounding up. Our architecture consumption estimation is shown in blue and compared to the cascaded GSS only in red and GSS + Synchronization for the DoG part in green. Note that our architecture does not require any synchronization in order to perform the adjacent scale subtraction since our scales are already in sync. Notice that we are able to use up to 9 times less BRAM resources compared to a cascaded architecture.

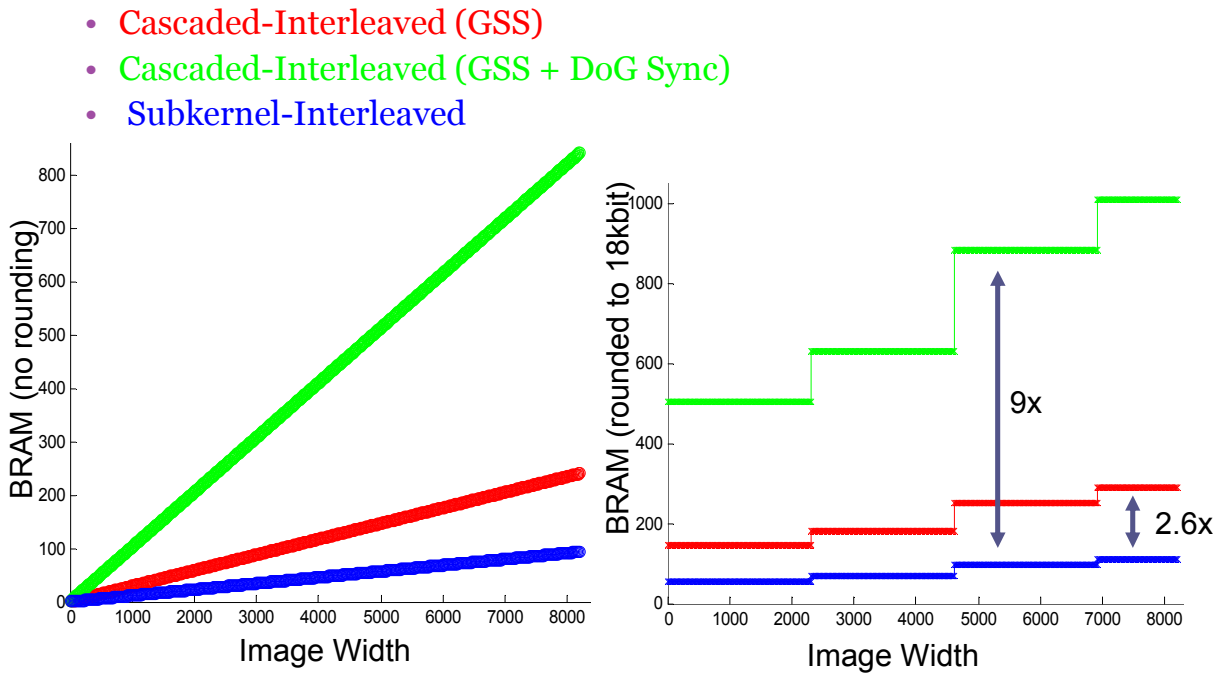


Figure 5.1 BRAM utilization estimate

As mentioned in Chapter 4, the architecture can be parameterized using generics to be

able to reconfigure the design according to the application needs. Also parametric sweeps were performed to show the effect on resource utilization and even accuracy as compared to a floating point software implementation. These also were compared to a cascaded interleaved architecture to highlight the difference and resource savings that can be achieved.

Moreover, it is to be noted that we also handled some finer details that are often ignored by other implementations such as providing the ability to perform different border handling methodologies such as the ones mentioned in Section 2.1.1.

In order to be able to implement the current architecture we also had to separate some modules into sub modules. Basically the FIR modules had to be separated into two major parts, a data buffer (multi-line buffer) or kernelizer and the actual filtering core. Moreover, the filtering was performed separately in the x and y directions. It is to be noted that the line buffers were implemented as cyclic RAM buffers as mentioned in Section 2.3.2 versus a direct shift register implementation which would not be suited for larger image widths.

Concerning the Huffman encoder, it was mainly implemented due to the high data redundancy caused by the Gaussian blurring. Compared to a direct Paeth predictor Paeth (1991) which uses the spatial image gradient to provide the best prediction, our modified Paeth predictor also incorporates scale information. This allows our differentiator, prior to the Huffman encoding, to have an even steeper histogram concentrated around 0.

Moreover, from our tests, it was shown that the Huffman table that was used did not have much of an impact on the compression results, except if the training images were artificially produced. An averaged table calculated from the Caltech-256 database proved to be generic enough.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

The Cortivision team is actively developing an intra-cortical implant to stimulate the visual cortex and provide visually impaired persons with basic vision. Our main target was to process images in order to "understand" image content and be able to simplify data by the means of a real-time embedded object recognition system.

We mainly contributed by introducing what we call the sub-kernel trick for the DoG implementation in order to save/share FPGA resources efficiently while still maintaining the precision and increasing the processing speed of the by two orders of magnitude compared to a software implementation. We also provided a methodology of using a modified Paeth predictor combined with a parallel Huffman encoder architecture to compress image pyramids as well as standard images to save on transmission bandwidth. Moreover, we suggested an alternative way that may further be explored for phosphene map estimation based on tracking fiducial markers using image processing tools.

6.1 Research Synthesis

In order to "simplify" an incoming video stream into a form that can be better transmitted to a patient, we focused on implementing an object recognition module that can operate in real-time.

Having reviewed the literature for various approaches as was discussed in Chapter 2, we were able to adopt the SIFT algorithm as a robust candidate to be used. The algorithm proved to be robust enough to handle different image variations such as brightness, contrast, scaling, rotation, and affine transformations.

A software prototype was implemented as a multithreaded application that merges the SIFT algorithm for recognition as discussed in Section 3.1. The SIFT is not able to operate in real-time for video processing on a host machine since it requires approximately one second to process a single frame and detect an object. The tracker was used to overcome that limitation by tracking the recognized object in real-time.

After identifying the Difference of Gaussians (DoG) as being the bottleneck of the SIFT algorithm, we moved on to migrating that part to an FPGA implementation. The DoG is mainly focused around repetitive Gaussian filtering and subsampling in order to the be able to detect stable keypoints through non-maxima suppression across scales. The resulting output

of the DoG is an image pyramid as the one shown in Figure 2.11.

The Xilinx System Generator was evaluated, in Section 3.2.1, as a potential tool for implementing the DoG system. Albeit being user friendly due to its block diagram nature, System Generator lacks the flexibility of performing generate loops such as the ones found in the VHDL language.

We focused on coding a highly parameterizable architecture that will be able to target different application needs, such accuracy vs. resource usage tradeoffs and did not hardcode any of the parameters. Moreover the VHDL was coded to be reusable and portable by using behavioral models where applicable and inferring instead of instantiating lower level blocks such as BRAM, and dedicated multipliers.

By acknowledging a stream processing data flow, we reviewed the various image processing techniques for FPGA implementations as shown in Section 2.3 and focused on having a proper data flow control.

In order to achieve a resource efficient design, our DoG implementation uses various concepts such as : pipelining, octave data interleaving, filter separability, filter symmetry, and the sub-kernel trick that we introduced.

The sub-kernel trick, that is explained in Section 3.2.3 was the contribution that gave us an advantage of the direct implementations and cascaded implementations. It basically relies on sharing BRAM resources for the vertical filtering part of the algorithm.

Parameter sweeps (width, taps, octaves, scales) of the architecture were performed and graphed as shown in Figure 4.13 and also compared to other architectures, see Figure 4.12, for a comparison of the FPGA resource estimation.

The hardware DoG implementation was also proven to be comparably accurate to a software floating point implementation.

The second part of the SIFT algorithm consists of calculating the keypoint descriptor around the local extrema that are found by the DoG part. This part relies on floating point calculations and is typically best offloaded to a host or dedicated sequential processor.

By noticing the data redundancy that is introduced by Gaussian blurring and the scale-space formation, as explained in Section 3.2.4, we noticed that the data is readily compressible and can save transmission bandwidth bottlenecks. We hence came up with a causal modified Paeth predictor that will take into account the spatial similarities as a well the scale similarities.

Furthermore, we introduced a new architecture for a parallel Huffman encoder which can be used for both image pyramid as well as simple image compression and showed lossless compression results of 27.3% on the Caltech-256 image dataset as shown in Table 4.4.

Our object recognition system is dedicated to a visual intra-cortical stimulator, hence the

final recognition result will have to be modulated to fit the existing phosphene map in the patient's Field of View (FOV). We thus also investigated a new methodology of performing phosphene map calibration based on a simple setup that employs a camera filming a table with fiducial markers that can be manipulated by the patient for direct real-time calibration feedback. Image processing tools such as the ones described in Section 4.4 were used to make that possible.

6.2 Future Recommendations

Due to the octave data interleaving, the DoG module is able to process 1 pixel per 2 clock cycles. This limitation can be overcome by processing several pixels at once, some of the implemented modules already support this feature.

Another limitation lies in the Huffman tables. Ideally the Huffman encoder should be able to form a new table by training itself on the incoming pixel stream and/or use adaptive tables.

Some future improvements may incorporate implementing the description formation and matching part of the SIFT algorithm on a processor such as the ARM that is present in the relatively new Xilinx Zynq family.

The SIFT features can also be used for scene recognition as stated in Lazebnik *et al.* (2006). This information can be integrated into the system to change the object search database depending on the context, for example if the patient is in a kitchen the system would poll a database of household items, whereas if he is outdoors, a database of cars, bikes, and stop signs may be polled. This will reduce false positive matches and useless comparisons.

Finally, the phosphene mapping technique can be further tested with patient feedback to incorporate a set of training patterns e.g. basic geometric shapes and moving objects. An emulator can be setup with virtual reality glasses to model the patient's FOV and the patient would manipulate the tracked fiducial markers to calibrate the perceived phosphene map in real-time.

6.3 Concluding Remarks

An object recognition system was presented. We showed how we can accelerate the DoG part of the SIFT algorithm in FPGAs while introducing the sub-kernel trick to implement resource efficient designs. Also, a parallel Huffman encoder preceded by a modified Paeth predictor was conceived for lossless image pyramid and image compression. Moreover, a new method of phosphene calibration was introduced that differs from the ones in the literature in that it relies on a camera setup combined with image processing techniques. Such a system

was conceived to be parameterizable and reusable for the purpose of providing the visually impaired with simplified vision, better navigational and even recognition abilities.

REFERENCES

ABRÀMOFF, M., MAGALHÃES, P. and RAM, S. (2004). Image processing with imagej. *Biophotonics international*, 11, 36–42.

ADELSON, E., ANDERSON, C., BERGEN, J., BURT, P. and OGDEN, J. (1984). Pyramid methods in image processing. *RCA engineer*, 29, 33–41.

BAY, H., ESS, A., TUYTELAARS, T. and VAN GOOL, L. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110, 346–359.

BAY, H., TUYTELAARS, T. and GOOL, L. V. (2006). Surf : Speeded up robust features. *ECCV*. 404–417.

BEIS, J. and LOWE, D. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1000–1006.

BONATO, V., MARQUES, E. and CONSTANTINIDES, G. (2008). A parallel hardware architecture for scale and rotation invariant feature detection. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18, 1703 –1712.

BOSCH, A., ZISSERMAN, A. and MUOZ, X. (2007). Image classification using random forests and ferns. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. Ieee, 1–8.

BRADSKI, G. (2000). The opencv library. *Doctor Dobbs Journal*, 25, 120–126.

BRADSKI, G. and KAEHLER, A. (2008). *Learning OpenCV*. O'Reilly Media Inc.

BROWN, M. and LOWE, D. (2002). Invariant features from interest point groups. *British Machine Vision Conference, Cardiff, Wales*. 656–665.

BROWN, M. and LOWE, D. (2003). Recognising panoramas. *Proceedings of the Ninth IEEE International Conference on Computer Vision*. vol. 2, 5.

BRUNELLI, R. (2009). *Template matching techniques in computer vision*. Wiley Online Library.

BUFFONI, L.-X., COULOMBE, J. and SAWAN, M. (2005). Image processing strategies dedicated to visual cortical stimulators : A survey. *Artificial Organs*, 29, 658–664.

CASTLE, R. O., KLEIN, G. and MURRAY, D. W. (2010). Combining monoslam with object recognition for scene augmentation using a wearable camera. *Journal of Image and Vision Computing*, 28, 1548 – 1556.

CHAI, X., ZHANG, L., LI, W., SHAO, F., YANG, K. and REN, Q. (2008). Study of tactile perception based on phosphene positioning using simulated prosthetic vision. *Artificial organs*, 32, 110–115.

CHANG, L. and HERNÁNDEZ-PALANCAR, J. (2009). A hardware architecture for sift candidate keypoints detection. *Proceedings of the 14th Iberoamerican Conference on Pattern Recognition : Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Springer-Verlag, Berlin, Heidelberg, CIARP '09, 95–102.

CHAPMAN, K. (2000). Saving costs with the srl16e. *Xilinx techXclusive*.

CHEN, B., DACHILLE, F. and KAUFMAN, A. (1999). Forward image mapping. *Proceedings of the conference on Visualization'99 : celebrating ten years*. IEEE Computer Society Press, 89–96.

CHOW, A., CHOW, V., PACKO, K., POLLACK, J., PEYMAN, G. and SCHUCHARD, R. (2004). The artificial silicon retina microchip for the treatment of vision loss from retinitis pigmentosa. *Archives of ophthalmology*, 122, 460.

COULOMBE, J., SAWAN, M. and GERVAIS, J. (2007). A highly flexible system for microstimulation of the visual cortex : Design and implementation. *Biomedical Circuits and Systems, IEEE Transactions on*, 1, 258–269.

CRISPIN, A. and RANKOV, V. (2007). Automated inspection of pcb components using a genetic algorithm template-matching approach. *The International Journal of Advanced Manufacturing Technology*, 35, 293–300.

DALAL, N. and TRIGGS, B. (2005). Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Ieee, vol. 1, 886–893.

DOBELLE, W. and MLADEJOVSKY, M. (1974). Phosphenes produced by electrical stimulation of human occipital cortex, and their application to the development of a prosthesis for the blind. *The Journal of physiology*, 243, 553.

DOLJANU, A. and SAWAN, M. (2007). 3D shape acquisition system dedicated to a visual intracortical stimulator. *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, 1313–1316.

DUDA, R. and HART, P. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15, 11–15.

FISCHLER, M. and BOLLES, R. (1981). Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24, 381–395.

GARRAULT, P. and PHILOFSKY, B. (2005). Hdl coding practices to accelerate design performance. *Xilinx Xcell Journal*, 31–35.

GHANNOUM, R. and SAWAN, M. (2007). A 90nm cmos multimode image sensor intended for a visual cortical stimulator. *Microelectronics, 2007. ICM 2007. Internatonal Conference on*. IEEE, 179–182.

GOSLIN, G. (1995). A guide to using field programmable gate arrays (fpgas) for application-specific digital signal processing performance. *Xilinx Inc.*

HARRIS, C. and STEPHENS, M. (1988). A combined corner and edge detector. *Alvey vision conference*. Manchester, UK, vol. 15, 50.

HAWKES, G. (2005). Dsp : Designing for optimal results. high-performance dsp using virtex-4 fpgas.

HESS, R. (2010). An open-source siftlibrary. *Proceedings of the international conference on Multimedia*. ACM, 1493–1496.

HUMAYUN, M., WEILAND, J., FUJII, G., GREENBERG, R., WILLIAMSON, R., LITTLE, J., MECH, B., CIMMARUSTI, V., VAN BOEMEL, G., DAGNELIE, G. *ET AL.* (2003). Visual perception in a blind subject with a chronic microelectronic retinal prosthesis. *Vision Research*, 43, 2573–2581.

JONKER, R. and VOLGENANT, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38, 325–340.

JUAN, L. and GWUN, O. (2009). A comparison of sift, pca-sift and surf. *International Journal of Image Processing*, 3, 143–152.

KE, Y. and SUKTHANKAR, R. (2004). Pca-sift : A more distinctive representation for local image descriptors. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Ieee, vol. 2, II–506.

KIM, E. T., SEO, J. M., ZHOU, J. A., JUNG, H. and KIRN, S. J. (2004). A retinal implant technology based on flexible polymer electrode and optical/electrical stimulation. *Biomedical Circuits and Systems, 2004 IEEE International Workshop on*, S1/8–12–15.

LAGANIÈRE, R. (1998). Morphological corner detection. *Computer Vision, 1998. Sixth International Conference on*. IEEE, 280–285.

LAGANIÈRE, R. (2011). *OpenCV 2 computer vision application programming cookbook*. Packt Publ. Limited.

LAZEBNIK, S., SCHMID, C. and PONCE, J. (2006). Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. *Proceedings of the 2006 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2. IEEE Computer Society, Washington, DC, USA, CVPR '06, 2169–2178.

LINDEBERG, T. (1994). *Scale-space theory in computer vision*. Springer.

LINDEBERG, T. (1998). Feature detection with automatic scale selection. *International journal of computer vision*, 30, 79–116.

LINDEBERG, T. (1999). Principles for automatic scale selection. *Handbook on Computer Vision and Applications*, 2, 239–274.

LOWE, D. (1999). Object recognition from local scale-invariant features. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Ieee, vol. 2, 1150–1157.

LOWE, D. (2001). Local feature view clustering for 3d object recognition. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. IEEE, vol. 1, I–682.

LOWE, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60, 91–110.

LUCAS, B. and KANADE, T. (1981). An iterative image registration technique with an application to stereo vision. *Proceedings of the 7th international joint conference on Artificial intelligence*.

MATLAB (2010). *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts.

MEYER-BAESE, U. (2004). *Digital signal processing with field programmable gate arrays*. Springer Verlag.

MIKOLAJCZYK, K. and SCHMID, C. (2004). Scale & affine invariant interest point detectors. *International journal of computer vision*, 60, 63–86.

MIKOLAJCZYK, K. and SCHMID, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27, 1615–1630.

MONTEMERLO, M., THRUN, S., KOLLER, D. and WEGBREIT, B. (2002). Fastslam : A factored solution to the simultaneous localization and mapping problem. *Proceedings of the National conference on Artificial Intelligence*. Menlo Park, CA ; Cambridge, MA ; London ; AAAI Press ; MIT Press ; 1999, 593–598.

NEUBECK, A. and VAN GOOL, L. (2006). Efficient non-maximum suppression. *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. Ieee, vol. 3, 850–855.

PAETH, A. W. (1991). *Graphics Gems II*. Academic Press.

PALANKER, D., VANKOV, A., HUIE, P. and BACCUS, S. (2005). Design of a high-resolution optoelectronic retinal prosthesis. *Journal of neural engineering*, 2, S105.

PATEL, H. (2005). Synthesis and implementation strategies to accelerate design performance. *Xilinx White Paper*, 229.

SAKAGUCHI, H., KAMEI, M., FUJIKADO, T., YONEZAWA, E., OZAWA, M., CECILIA-GONZALEZ, C., USTARIZ-GONZALEZ, O., QUIROZ-MERCADO, H. and TANO, Y. (2009). Artificial vision by direct optic nerve electrode (av-done) implantation in a blind patient with retinitis pigmentosa. *Journal of Artificial Organs*, 12, 206–209.

SE, S., LOWE, D. and LITTLE, J. (2002). Global localization using distinctive visual features. *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on.* IEEE, vol. 1, 226–231.

SHI, J. and TOMASI, C. (1994). Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on.* IEEE, 593–600.

SRIVASTAVA, N., TROYK, P., TOWLE, V., CURRY, D., SCHMIDT, E., KUFTA, C. and DAGNELIE, G. (2007). Estimating phosphene maps for psychophysical experiments used in testing a cortical visual prosthesis device. *Neural Engineering, 2007. CNE '07. 3rd International IEEE/EMBS Conference on*, 130–133.

TODOROVIC, D. (1996). A gem from the past : Pleikart stumpf's (1911) anticipation of the aperture problem, reichardt detectors, and perceived motion loss at equiluminance. *PERCEPTION-LONDON-*, 25, 1235–1242.

TOLEDO, E., MARTINEZ, J., GARRIGOS, E. and FERRANDEZ, J. (2005). FPGA implementation of an augmented reality application for visually impaired people. *Field Programmable Logic and Applications, 2005. International Conference on*, 723–724.

VEDALDI, A. and FULKERSON, B. (2010). Vlfeat – an open and portable library of computer vision algorithms. *Proceedings of the 18th annual ACM international conference on Multimedia*. Firenze.

VERAART, C., WANET-DEFALQUE, M., GÉRARD, B., VANLIERDE, A. and DELBEKE, J. (2003). Pattern recognition with the optic nerve visual prosthesis. *Artificial organs*, 27, 996–1004.

VIOLA, P. and JONES, M. (2004). Robust real-time face detection. *International journal of computer vision*, 57, 137–154.

WINTER, J., COGAN, S. and RIZZO, J. (2007). Retinal prostheses : current challenges and future outlook. *Journal of Biomaterials Science, Polymer Edition*, 18, 1031–1055.

YANAI, D., WEILAND, J., MAHADEVAPPA, M., GREENBERG, R., FINE, I. and HUMAYUN, M. (2007). Visual performance using a retinal prosthesis in three subjects with retinitis pigmentosa. *American journal of ophthalmology*, 143, 820–827.

YILMAZ, A., JAVED, O. and SHAH, M. (2006). Object tracking : A survey. *Acm Computing Surveys (CSUR)*, 38, 13.

ZRENNER, E., BARTZ-SCHMIDT, K., BENAV, H., BESCH, D., BRUCKMANN, A., GABEL, V., GEKELER, F., GREPPMAIER, U., HARSCHER, A., KIBBEL, S. *ET AL.* (2011). Subretinal electronic chips allow blind patients to read letters and combine them to words. *Proceedings of the Royal Society B : Biological Sciences*, 278, 1489–1497.